U.S. DEPARTMENT OF COMMERCE
National Technical Information Service

AD-A030 147

Numerical Fourier Transform

Harry Diamond Labs Adelphi Md

Sep 76

AD A030147

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER HDL-TR-1748	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Numerical Fourier Transform		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Alfred G. Brandstein Egon Marx		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harry Diamond Laboratories 2800 Powder Mill Road Adelphi, MD 20783		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program: 6.21.18A DA: 1W162118AH75
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Materiel Development and Readiness Command Alexandria, VA 22333		12. REPORT DATE September 1976
		13. NUMBER OF PAGES 49
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

HDL Project: X755E2 DRCMS Code: 612118.11.H7500

This work was funded under DARCOM NWER/T Program element 1W162118AH75/C, titled Multiple Systems Evaluation Program.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Fourier transforms
Computer software
Digitized data

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Numerical techniques for the analysis of digitized time-amplitude traces are presented, with special emphasis on Fast Fourier transforms. Accuracy of computation is improved by the use of linearly interpolated functions; for equispaced input the subprogram FLAT uses the Cooley-Tukey algorithm, while NUFT may be employed for the nonequispaced case. Althought FLAT and NUFT can be used to perform inverse Fourier transforms, the special subprograms FLIT and INUFT compute these functions more

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED PRICES SUBJECT TO CHANGE
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

efficiently. An analysis of the errors introduced by digitization, interpolation, and the computation of Fourier transforms is included.

CONTENTS

		Page
1.	INTRODUCTION	7
2.	THE DISCRETE FOURIER TRANSFORM	8
3.	THE FAST FOURIER TRANSFORM	8
	3.1 Aliasing–Theorem 1 (Cooley)	9
	3.2 Applications of Theorem 1	10
	3.2.1 Choice of N and Δt for Reasonable Approximations	10
	3.2.2 Inverse Transforms	10
4.	SUBROUTINE FLAT	10
	4.1 Derivation of FLAT	11
5.	SUBROUTINES NUFT AND INUFT	13
6.	SUBROUTINES LINT AND CLINT	14
7.	MIMIPULSE	15
8.	SMOOTHING	17
9.	DIGITIZATION AND TRANSFORM ERRORS	18
10.	SUBROUTINES RDTAPE AND CSTOUT	31
	APPENDIX A. –LISTINGS OF PROGRAMS USED IN NUMERICAL FOURIER TRANSFORMS	33
	DISTRIBUTION	51

FIGURES

1.	Absolute values of analytic transform of Mimipulse and FLAT of equispaced Mimipulse for 256 points	19
2.	Absolute values of analytic transform of Mimipulse and FFT of equispaced Mimipulse for 256 points	20
3.	Absolute values of analytic transform of Mimipulse and FLAT of equispaced Mimipulse for 512 points	20
4.	Absolute values of analytic transform of Mimipulse and FFT of equispaced Mimipulse for 512 points	21

Page

5. Absolute values of analytic transform of Mimipulse and FLAT (first 90 MHz) of equispaced Mimipulse for 1024 points 21

6. Absolute values of analytic transform of Mimipulse and FFT (first 90 MHz) of equispaced Mimipulse for 1024 points 22

7. Absolute values of analytic transform of Mimipulse and FLAT of equispaced Mimipulse for 2048 points. 22

8. Absolute values of analytic transform of Mimipulse and FFT of equispaced Mimipulse for 2048 points 23

9. First 90 MHz of analytic transform of Mimipulse and FLAT of digitized Mimipulse 2048 points interpolated array 23

10. Analytic transform of Mimipulse and FLAT of digitized Mimipulse 2048 point interpolated array 24

11. Analytic transform of Mimipulse and FLAT of array whose values are analytic values of Mimipulse at digitized time points interpolated to 2048 points to simulate sampling errors. 24

12. FLAT of analytic values of Mimipulse evaluated at digitized time points, quantified to multiples of 1/1000 of maximum value and linearly interpolated to 2048 points, simulating sampling and quantification errors and analytic transform of Mimipulse 25

13. FLAT of analytic values of Mimipulse evaluated at digitized time points quantified to multiples of 1/1000 of maximum value and R multiples of 1/1000 of maximum value, where R is random number between -10 and 10, simulating effects of digitization and analytic transform of Mimipulse 25

14. First 90 MHz of graph in figure 13 26

15. Mimipulse and FLIT of analytic transform of Mimipulse for 256 points. 26

16. Mimipulse and double aliased FFT of analytic transform of Mimipulse for 256 points, with value at 0 subtracted 27

17. Mimipulse and single aliased FFT of the analytic transform of Mimipulse for 256 points, with value at 0 subtracted 27

18. Mimipulse and FLIT of analytic transform of Mimipulse for 1024 points. 28

19. Mimipulse and double aliased FFT of analytic transform of Mimipulse for 1024 points. 28

Page

20. Mimipulse and single aliased FFT of analytic transform of Mimipulse for 1024 points 29

21. FLIT of analytic transform of $k\left(e^{\alpha t} - e^{\beta t}\right)$ for 256 points and $k\left(e^{\alpha t} - e^{\beta t}\right)$ . . 29

22. Graph of $k\left(e^{\alpha t} - e^{\beta t}\right)$ and double aliased FFT of analytic transform of $k\left(e^{\alpha t} - e^{\beta t}\right)$ for 256 points with value at 0 subtracted 30

23. Graph of $k\left(e^{\alpha t} - e^{\beta t}\right)$ and single aliased FFT of analytic transform $k\left(e^{\alpha t} - e^{\beta t}\right)$ with value at 0 substracted 30

1. INTRODUCTION

Much time and effort have been expended in attempting to approximate the Fourier transform of functions. The Fourier transform, h, of a function g is given by

$$h(f) = \int_{-\infty}^{\infty} g(t)e^{p2\pi ift}dt, \ p = \pm 1. \qquad (1)$$

In addition, for reasonable functions, the inverse to this transform is given by

$$g(t) = \int_{-\infty}^{+\infty} h(f)e^{-p2\pi ift}df. \qquad (2)$$

For this report, let g be real and continuous and vanish outside the interval [0,T]. In this case, h(f) is continuous, and g is the inverse transform of h. The definition of the Fourier transform may easily be extended to distributions of which the Dirac δ function is an example.

This report discusses four different transforms that under certain conditions may be considered approximations to the Fourier transform. The Discrete Fourier Transform (DFT) and the Fast Fourier Transform (FFT) are well known and represent the transform of impulses; the other two are the transforms of piece-wise linear functions and are implemented by the subprograms FLAT and NUFT.

The DFT replaces the original function, g, with N impulses (point masses at N equispaced points in [0, T] weighted by the value of g at these points). The Fourier transform of the sum of these impulses is then computed. Thus, the DFT has a value at each frequency point. The FFT produces a sampling of the DFT at N equispaced frequency points. The FFT makes use of the Cooley-Tukey algorithm to calculate these values rapidly and efficiently.

Subroutine NUFT is an algorithm that calculates the Fourier transform of virtually any piece-wise linear function. If the endpoints of these linear segments are equispaced, N (the number of points) is a power of 2, and the function vanishes outside the interval, subroutine FLAT may be used to sample NUFT at N equispaced frequency points. Subroutine FLAT makes use of the Cooley-Tukey algorithm to produce these results in about the same computer time as an FFT.

Since each of these transforms is the Fourier transform of an approximation to the original function, the degree to which each agrees with the Fourier transform is largely determined by the goodness of fit to the original functions.

Since it is of great interest to apply these transforms to the analysis of rapidly varying transient signals, this report discusses also the routines used to process these signals at the Harry Diamond Laboratories (HDL). Also discussed is Mimipulse--a function used to simulate the response of systems to electromagnetic pulse (EMP)--as well as the calculation of its transform. Mimipulse was often employed as a means of testing the validity of the various processes.

The programs are listed in appendix A.

2. THE DISCRETE FOURIER TRANSFORM

Let $\Delta t > 0$ and N be a positive integer; then the DFT of g of order N and increment Δt is given by

$$h(f) = \sum_{j=0}^{N-1} g(j\Delta t)e^{p2\pi i f\Delta t}, p = \pm 1 . \tag{3}$$

This transform is of interest because it and its inverse are relatively easy to compute.

Insight into the DFT may be gained by observing that if we define

$$g_1(t) = \sum_{j=0}^{N-1} g(j\Delta t)\,\delta(t - j\Delta t) , \tag{4}$$

where δ = Dirac delta function. That is, g_1 is a sum of impulses at the points $\left\{j\Delta t\right\}_{j=0}^{N-1}$, and the DFT of g is identical to the Fourier transform of g_1. Also, the DFT defines a mapping from functions defined on a discrete set,

$$\left\{j\Delta t\right\}_{j=0}^{N-1} ,$$

to functions defined for all f, $-\infty < f < \infty$, for if two functions agree on this discrete set, their DFT's agree everywhere.

3. THE FAST FOURIER TRANSFORM

The FFT for certain N is an extremely fast method of sampling the DFT. The heart of the FFT is the Cooley-Tukey algorithm, which is a very efficient means of summing the series

$$\sum_{j=0}^{N-1} a_j e^{p2\pi ijk/N}, p = \pm 1 \tag{5}$$

for integer k, when N is a composite number. The Cooley-Tukey algorithm has highest efficiency when $N = 2^m$. In that case, the ratio of computer time spent summing by use of this algorithm, as opposed to more conventional means, is approximately m/N. The Cooley-Tukey algorithm is applied to the DFT by the observation that equation (3) reduces to

$$h(k\Delta f) = \sum_{j=0}^{N-1} g(j\Delta t)e^{p2\pi ijk/N} \tag{6}$$

if f is replaced by $k\Delta f$, where $\Delta f = 1/(N\Delta t)$.

The output from the FFT is an ordered array of N complex numbers. The first (N/2 + 1) numbers of the array represent the values at frequencies in $[0, (N/2)\Delta f]$ inclusive while the rest of the array represents the values at frequencies in $[(-N/2 + 1)\Delta f, -\Delta f]$ inclusive. Thus, in this array, the negative frequency values follow the positive ones, and the highest frequency represented is $(N/2)\Delta f$. For a real function g, the value at a negative frequency is the complex conjugate of the value at the corresponding positive frequency.

3.1 Aliasing–Theorem 1 (Cooley)

The key to understanding aliasing is a remarkable theorem of Cooley's.[1] Let $\Delta f = 1/(N\Delta t) = 1/T$, $F = N\Delta f$. Define

$$g_p(t) = \sum_{\ell=-\infty}^{\infty} g(t + \ell T)\,, \; 0 \leq t < T\,, \tag{7}$$

$$h_p(f) = \sum_{m=-\infty}^{\infty} h(f + mF)\,, \; -\frac{F}{2} < f \leq F/2\,. \tag{8}$$

Theorem 1 (Cooley).–If h is the Fourier transform of g, then h_p is the FFT of g_p (where the appropriate sampling is made).

Definition.–Aliasing is that error introduced into the FFT by the contribution of frequencies higher than those considered. Observe that

a. If $g(t) = 0$ for $t \geq T$ and $t < 0$, then $g_p(t) = g(t)$. Thus, the FFT of g agrees with h_p sampled at

$$\left\{k\Delta f\right\}_{N/2+1}^{N/2}\,.$$

In this case, the FFT evaluated at $m\Delta f$ differs from the Fourier transform, h, evaluated at $m\Delta f$ by

$$\sum_{\ell \neq 0}^{\infty} h(m\Delta f + \ell F)\,.$$

b. If $h(f) = 0$ for $|f| \geq F/2$, then $h_p(k\Delta f)$ agrees with $h(k\Delta f)$ for $N/2 + 1 \leq k \leq N/2$. In this case, the inverse FFT sampled at $m\Delta t$, $0 \leq m \leq N$ differs from $g(m\Delta t)$ by

$$\sum_{\ell \neq 0}^{\infty} g(\Delta t + \ell T)\,.$$

[1] *J. W. Cooley, P.A.W. Lewis, and P. D. Welch, The Finite Fourier Transform, IEEE Trans. Audio Electroacoust.,* 17 *(June 1969), 77-85.*

c. If both the conditions for "a" and "b" are met, then the FFT agrees with the Fourier transform at the specified points. Unfortunately, the only allowable function satisfying these criteria is **$g \equiv 0$ (since a nonzero continuous function cannot be both band limited and time limited).**

3.2 Applications of Theorem 1

3.2.1 Choice of N and Δt for Reasonable Approximations

Given g, choose S large enough so that g(t) is negligible for $t > S$, and choose N large enough so that h(f) is negligible for $|f| > F/2$ where $F = N/S$. Then let $\Delta t = S/(N - 1)$. In this case, the FFT is a reasonable approximation to the Fourier transform sampled at these points.

3.2.2 Inverse Transforms

If one is given equispaced samples of the Fourier transform and wishes to use the FFT to evaluate the time-domain function, one should try to duplicate h_p as closely as possible by single or double aliasing.

Single Aliasing.—If equispaced samples are available merely in the range [0, F/2] , construct an equispaced array whose second half is the conjugate of the reflection about F/2.

In FORTRAN notation, single aliasing is performed by

```
      N1 = N/2+1
      N2 = N1-2
      F(N1) = 2*REAL(F(N1))
      DO 1  I = 1, N2
    1 F(N1 + I) = CONJG(F(N1-I)) .
```

Double Aliasing.—If equispaced samples are available in the frequency range [0, F] , replace the second half of the array with the sum of the original value and the conjugate of the reflection about F/2. The first half of the array is then replaced by the reflection of the new second half. Double aliasing may be achieved in FORTRAN by

```
      DO 2  I = 1, N2
      F(N1 + I) = F(N1 + I) + CONJG(F(N1-I))
    2 F(N1-I) = CONJG(F(N1 + I)) .
```

For both single and double aliasing, sufficient information is often not available to completely determine the best 0-frequency value to use. In the inverse transform, this can lead to either drift of the time-domain function or errors for small values of t. Thus, often the analyst anchors the transform by subtracting from the points of the time array the value at 0 time. Doing so, however, may introduce a relative displacement of the resulting time function, often noticeable at late times.

4. SUBROUTINE FLAT

In light of the inherent deficiencies of the FFT pointed out by theorem 1 and the nature of the digitization process employed by HDL, an alternate fast approximation to the Fourier transform was desired.

This transform was named "FLAT". It approximates the given function, g, by a piece-wise linear function, g_2, and then uses the Cooley-Tukey algorithm to obtain a sampling of the Fourier transform of g_2 at the same frequency points as does the FFT.* Approximation of continuous functions by line segments is inherently better for integration than approximation by impulses as employed by the FFT. Computer running times for FLAT are within 25 percent of those for the FFT (approx 200 ms for a 2048-point complex array on a Control Data Corporation (CDC) 6600 computer).

4.1 Derivation of FLAT

Consider a function g(t) defined on the interval $[0, (N-1)\Delta t]$ with $g(0) = 0$; extend g to the point $(N\Delta t, 0)$ linearly. Let $g_2(t)$ be the piece-wise linear function joining $\big(j\Delta t, g(j\Delta t)\big)$ to $\big((j+1)\Delta t, g[(j+1)\Delta t]\big)$, $j = 0, N-1$:

$$g_2(t) = \bigcup_{j=0}^{N-1} g_{2j}(t) ,$$

where

$$g_{2j}(t) = c_j t + d_j ,$$

$$t \in [j\Delta t, (j+1)\Delta t] .$$

$$c_j = \frac{g(j+1)\Delta t - g(j\Delta t)}{\Delta t} .$$

Then the Fourier transform, G_2, of g_2 is given (for $f \neq 0$) by

$$G_2(f) = \int_{-\infty}^{\infty} g_2(t) e^{(p2\pi ift)}\, dt = \int_{o}^{N\Delta t} g_2(t) e^{(p2\pi ift)}\, dt$$

$$= \sum_{j=0}^{N-1} \int_{j\Delta t}^{(j+1)\Delta t} g_j(t)\, e^{(p2\pi ift)}\, dt$$

$$= \sum_{j=0}^{N-1} \frac{-c_j \left[e^{(p2\pi if(j-1)\Delta t)} - e^{(p2\pi ifj\Delta t)}\right]}{(2\pi if)^2}$$

$$= \frac{1}{(2\pi if)^2} \sum_{j=0}^{N} s_j e^{(p2\pi if\Delta t)} .$$

**The Cooley-Tukey algorithm used is another one of the brilliant creations of W.T. Wyatt of HDL, written in COMPASS.*

This transform was named "FLAT". It approximates the given function, g, by a piece-wise linear function, g_2, and then uses the Cooley-Tukey algorithm to obtain a sampling of the Fourier transform of g_2 at the same frequency points as does the FFT.* Approximation of continuous functions by line segments is inherently better for integration than approximation by impulses as employed by the FFT. Computer running times for FLAT are within 25 percent of those for the FFT (approx 200 ms for a 2048-point complex array on a Control Data Corporation (CDC) 6600 computer).

4.1 Derivation of FLAT

Consider a function g(t) defined on the interval $[0, (N-1)\Delta t]$ with $g(0) = 0$; extend g to the point $(N\Delta t, 0)$ linearly. Let $g_2(t)$ be the piece-wise linear function joining $\left(j\Delta t, g(j\Delta t)\right)$ to $\left((j+1)\Delta t, g[(j+1)\Delta t]\right)$, $j = 0, N-1$:

$$g_2(t) = \bigsqcup_{j=0}^{N-1} g_{2j}(t) ,$$

where

$$g_{2j}(t) = c_j t + d_j ,$$

$$t \epsilon [j\Delta t, (j+1)\Delta t] ,$$

$$c_j = \frac{g(j+1)\Delta t - g(j\Delta t)}{\Delta t} .$$

Then the Fourier transform, G_2, of g_2 is given (for $f \neq 0$) by

$$G_2(f) = \int_{-\infty}^{\infty} g_2(t)e^{(p2\pi ift)} \, dt = \int_0^{N\Delta t} g_2(t)e^{(p2\pi ift)} \, dt$$

$$= \sum_{j=0}^{N-1} \int_{j\Delta t}^{(j+1)\Delta t} g_j(t)_e{}^{(p2\pi ift)} \, dt$$

$$= \sum_{j=0}^{N-1} \frac{-c_j \left[e^{(p2\pi if(j-1)\Delta t)} - e^{(p2\pi ifj\Delta t)}\right]}{(2\pi if)^2}$$

$$= \frac{1}{(2\pi if)^2} \sum_{j=0}^{N} s_j e^{(p2\pi if\Delta t)} ,$$

*The Cooley-Tukey algorithm used is another one of the brilliant creations of W.T. Wyatt of HDL, written in COMPASS.

where

$$s_o = c_o ,$$

$$s_N = c_{N-1} ,$$

$$s_j = c_j - c_{j-1} , 0 < j < N - 1 .$$

Now let $\Delta f = 1/(N\Delta t)$:

$$G_2(k\Delta f) = \frac{1}{(2\pi ik\Delta f)^2} \sum_{j=0}^{N} s_j e^{\left(\frac{p2\pi ijk}{N}\right)}$$

$$= \frac{1}{(2\pi ik\Delta f)^2} \sum_{j=0}^{N-1} t_j e^{\left(\frac{p2\pi ijk}{N}\right)},$$

where

$$t_o = c_o - c_{N-1} ,$$

$$t_j = s_j$$

for $j > 0$.

Next, observe that

$$\sum_{j=0}^{N-1} t_j e^{\left(\frac{p2\pi ijk}{N}\right)}$$

may be evaluated by use of the Colley-Tukey algorithm.

It is of interest to observe that if $|g(t) - g_2(t)| < \epsilon, 0 \leqslant t < T$, then

$$\left| \int_o^T g(t)e^{(p2\pi ift)} dt - \int_o^T g_2(t)e^{(p2\pi ift)} dt \right| \leqslant \int_o^T |g(t) - g_2(t)| dt < \epsilon T$$

independent of f. This implies that, if g(t) is continuous and vanishes for $t > T$, the Fourier transform of g_2 converges uniformly to the Fourier transform of g as $\Delta t \to 0$. Hence, the values of FLAT converge to the values of the Fourier transform at the sampled points.

An inverse to FLAT, called "FLIT," also uses the Cooley-Tukey algorithm. Subroutine FLIT produces an equispaced time array,

$$\{g|k\Delta t\}_{k=0}^{N-1}$$

with $g(0) = 0$, such that FLAT of the (linearly interpolated) array is the given equispaced frequency array.

Indeterminacy of the 0-frequency value may result in a tilt of the time-domain curve or oscillations for large values of t.

Subroutines FLAT and FLIT are resident in the user library ANAPAC on the CDC 6600 computer at Fort Belvoir, VA.

The calling sequence for FLAT is:

Call FLAT (ARRAY, N, DT, I),

where

ARRAY	=	name of complex array containing data,
N	=	number of points in array (N must be a power of 2),
DT	=	Δt,
I	=	value of p desired in transform.

The calling sequence for FLIT is

Call FLIT(ARRAY, N, DF, I),

where

ARRAY	=	name of complex array containing data (only the first N/2 + 1 points need be specified); the desired output is the real part of ARRAY,
N	=	number of points in array (N must be a power of 2),
DF	=	Δf,
I	=	value of p desired in inverse transform.

The desired output is the real part of ARRAY.

5. SUBROUTINES NUFT AND INUFT

Subroutines NUFT and INUFT are used to compute direct and inverse Fourier transforms.

The arguments in the CALL NUFT card are

(1) Real array: independent input variable (time)
(2) Real array: dependent input variable (amplitude)
(3) Integer: number of points in input arrays
(4) Integer: number of points in output arrays
(5) Real array: independent output variable (frequency)
(6) Integer: indication of whether the array in No. 5 is given as frequency or circular frequency and whether it remains that way or changes (see comments for IOM)
(7) Complex array: dependent output variable (Fourier transform)
(8) Real array: storage for intermediate results to save computation time, dimensioned as the input arrays
(9) Integer: sign of the exponential (+1 or −1).

Terms with small absolute value in the exponential are computed by use of a power series expansion, to avoid loss of accuracy due to subtraction of almost equal numbers.

The Fourier transform is computed assuming that the time-amplitude trace is formed of straight line segments.

It is possible to use NUFT to perform an inverse Fourier transform by applying it separately to the real and imaginary parts of the given transform. This operation takes approximately twice as long as the direct Fourier transform. In INUFT, those operations are performed only once, just as in NUFT. The call is quite similar to that of NUFT, but arrays No. 1 and 2 are for the output, arrays No. 5 and 7 are for the input, and the scratch array No. 8 has to be complex and of dimension given by the integer in argument No. 4. The values of the Fourier transform are given only for nonnegative frequencies, and the output function is assumed to be real.

These transforms have to be used when it is not convenient to have equispaced input and output points in equal numbers, usually a power of 2. It is often important to use frequency points with intervals that increase with increasing frequency, to sample closely the low-frequency values, and to go to very high frequencies at least with a few points. The results of inverse Fourier transforms are improved by this procedure.

6. SUBROUTINES LINT AND CLINT

Subroutines LINT and CLINT are used to interpolate linearly between points in a given array. The arguments in LINT are

(1) Real array: independent input variable
(2) Real array: dependent input variable
(3) Integer: number of points in input arrays
(4) Integer: number of points in output array
(5) Real: increment for independent output variable
(6) Real array: dependent output variable.

The independent output variable starts with the first value of the input array. If any points fall outside the range of the input variable, a value of 0 is given to the output.

Subroutine CLINT differs from LINT only in the sixth argument, which is a complex array.

7. MIMIPULSE*

A model that has been used to represent analytically the response to an electromagnetic pulse is a function that vanishes for $t < 0$ and is defined by

$$F(t) = \begin{cases} At/T_1 & 0 \leqslant t \leqslant T_1 \\ A \exp[-\xi_1(t - T_1)] \cos\left[\dfrac{\pi(t - T_1)}{2(T_2 - T_1)}\right], & T_1 \leqslant t \leqslant T_2 \\ \alpha + \beta t + \delta t^2 + \gamma t^3, & T_2 \leqslant t \leqslant T_3 \\ a \exp[-\xi_2(t - T_3)] \cos[\omega_1(t - T_3)] & \\ \quad + (a\xi_2/\omega_2) \exp[-\xi_3(t - T_3)] \sin[\omega_2(t - T_3)], & T_3 \leqslant t \leqslant T_E \\ 0, & t > T_E . \end{cases}$$

The values of A, T_1, T_2, T_3, and a are given. In addition, a time, T_E, is given to define the end of the "recorded" part of the pulse. The values of ξ_1, ξ_2 and ξ_3 are given through constants S_1, S_2, and S_3 by

$$\xi_1 = \frac{1}{S_1(T_2 - T_1)},$$

$$\xi_2 = \frac{1}{S_2(T_E - T_3)},$$

$$\xi_3 = \frac{1}{S_3(T_E - T_3)},$$

and the circular frequencies are given by

$$\omega_1 = \frac{\pi}{2(T_4 - T_3)},$$

$$\omega_2 = \frac{\pi S_4}{T_E - T_3}.$$

Finally, α, β, δ, and γ are determined by matching the values and the derivatives of the function at the ends of the interval.

*A concept originally developed by Carl Konschnik formerly of HDL.

The Fourier transform of this function is the sum of the Fourier transforms of the components in the different intervals:

$$F_1(\omega) = \frac{A\left[\exp(-i\omega T_1)(i\omega T_1 + 1) - 1\right]}{T_1 \omega^2} ,$$

$$F_2(\omega) = A \exp(-i\omega T_1) \left[(\xi_1 + i\omega)^2 + \frac{\pi^2}{4(T_2 - T_1)^2}\right]^{-1}$$

$$\times \left\{ \exp[-(\xi_1 + i\omega)(T_2 - T_1)] \frac{\pi}{2(T_2 - T_1)} + (\xi_1 + i\omega)\right\} ,$$

$$F_3(\omega) = \exp(-i\omega T_3) \left[\frac{a}{-i\omega} + \frac{2\delta + 6\gamma T_3}{i\omega^3} - \frac{6\gamma}{\omega^4}\right] - \exp(-i\omega T_2) \left[\frac{Y}{\omega^2} + \frac{2\delta + 6\gamma T_2}{i\omega^3} - \frac{6\gamma}{\omega^4}\right] ,$$

$$F_4(\omega) = a \exp(-i\omega T_3) \left[\frac{\xi_2 + i\omega}{(\xi_2 + i\omega)^2 + \omega_1^2}\right.$$

$$+ \frac{\xi_2}{(\xi_3 + i\omega)^2 + \omega_2^2} + \eta \frac{\exp[-(\xi_2 + i\omega)(T_E - T_3)]^2}{(\xi_2 + i\omega)^2 + \omega_1^2}$$

$$\times \left[-(\xi_2 + i\omega) \cos \omega_1(T_E - T_3) + \omega_1 \sin \omega_1(T_E - T_3)\right]$$

$$+ \eta \frac{\xi_2}{\omega_2} \frac{\exp\left[-(\xi_3 + i\omega)(T_E - T_3)\right]}{(\xi_3 + i\omega)^2 + \omega_2^2}$$

$$\left. \times \left[-(\xi_3 + i\omega) \sin \omega_2(T_E - T_3) - \omega_2 \cos \omega_2(T_E - T_3)\right] \right] ,$$

where

$$\gamma = -\frac{2a}{(T_3 - T_2)^2} + \frac{Y}{(T_3 - T_2)^3} ,$$

$$Y = -\frac{\pi A}{2(T_2 - T_1)} \exp[-\xi_1(T_2 - T_1)] ,$$

$$\delta = -\frac{a}{(T_3 - T_2)^2} - \gamma(T_2 + 2T_3) ,$$

and η is 1 or 0, depending on whether one assumes that the trace vanishes after T_E or is given by the expression in the last interval.

Functions F_1 and F_3 have to be computed separately for $\omega = 0$. They become

$$F_1(0) = \frac{AT_1}{2},$$

$$F_3(0) = \alpha(T_3 - T_2) + \frac{\beta(T_3^2 - T_2^2)}{2} + \frac{(T_3^3 - T_2^3)}{3} + \frac{\delta(T_3^4 - T_2^4)}{4},$$

where

$$\alpha = -T_2\beta - T_2^2\delta - T_2^3\gamma,$$

$$\beta = -2T_3\delta - 3T_3^2\gamma.$$

Three characteristics of the function follow: (1) It is continuous at the extremes of the intervals, but it does not necessarily vanish at T_E. (2) The derivative is discontinuous at T_1, but continues at T_2, where the function vanishes, and at T_3, where the function reaches a minimum given by a. (3) At T_4, the first term in the corresponding expression for f(t) vanishes.

Subroutines ANMI and ANTRA compute the values of f(t) and its Fourier transform. Two real arrays are passed to ANMI, one for the input (times) and one for the output (amplitudes), plus an integer that gives the number of points. Also, the subroutine reads a NAMELIST card PULSE that contains the values A, AA, T1, T2, T3, T4, TE, S1, S2, S3, and S4; the meanings are obvious, with the possible exception of AA = a. A real array for the input (circular frequencies) and a complex array for the output (values of the Fourier transform) are passed to ANTRA, with an integer for the number of points. This subroutine also reads a NAMELIST card with the name PULSE. It contains the same information as that for ANMI, plus a real variable FINIT that takes the value 0. for an infinite trace and 1. for a finite one.

8. SMOOTHING

The power spectrum obtained from a digitized time-amplitude trace shows a strong oscillating noise component, especially on a logarithmic scale. Subroutine SMUZ can be used to present the output in a more intelligible form. The input is an array of a function given at constant intervals. When two maxima or two minima occur closer than a prescribed number of points, the function in between is replaced by an average value between the straight lines joining the maxima and those joining the minima, with a tapered beginning and end. The process is repeated until there are no changes in a complete pass. Two different thresholds can be prescribed for two sectors of the function. The number of passes through the procedure is printed in the output; a number of four to six passes is usual. The computation of the average ordinate is performed by the subroutine AVRG called by SMUZ. The parameters in CALL SMUZ are:

(1) Real array: ordinates of the function being smoothed
(2) Integer: number of points in the array
(3) Integer: threshold number of points for the first part of the curve
(4) Integer: threshold number of points for the second part of the curve
(5) Real: fraction of points in the first part of the curve.

The thresholds have to be chosen so that the unwanted noise is eliminated while the significant extrema remain; a number to start with might be 1/100 of the total number of points.

It has been found that the use of this subroutine on the real and imaginary parts of the Fourier transform that subsequently has to be inverted tends to introduce spurious oscillations for late times.

9. DIGITIZATION AND TRANSFORM ERRORS

Digitization is routinely performed at HDL, with a Science Accessories Corp. GP2 digitizer, which allows for rear enlarged projection of oscillograms. The tablet is 20 by 20 in. and has a definition of 0.01 in. The output of this machine is cards punched on an 026 keypunch. Normally, recordings of a particular event consist of four or more traces at differing sweep speeds. Further processing of the data (e.g., time tying, sequence checking) is performed on a CDC 6600 computer with a complete and innovative software package developed by T.V. Noon of HDL (unpublished).

To evaluate the effects caused by the digitization and transform process, numerous experiments were performed using Mimipulse (sect. 7). Plots of Mimipulse were constructed by the same instructions to the operator as would be used for oscillograms processing (e.g., marking just the endpoints of apparent straight lines).

Several different results are presented (all compared to the analytic transform of Mimipulse):

a. Transforms of equispaced Mimipulse—to assess transform errors (fig. 1-8).

b. Transforms applied to the digitized points—to determine the combined errors due to digitization and transform (fig. 9, 10).

c. Transforms applied to the array whose values are the actual values of Mimipulse at the digitized time points—to simulate sampling errors (fig. 11).

d. Transforms applied to the array whose values are the truncation of the analytic values at the digitized time points—to simulate quantification errors (fig. 12).

e. Transforms applied to the array whose values are the truncation of the analytic values plus a random number (between −10 and 10) of points on the digitization tablet—to simulate the effects of the digitization-transform process (fig. 13, 14).

One major conclusion that can be drawn from this set of experiments: At least for oscillograms of traces similar to Mimipulse, errors introduced by the digitization-transformation process are minor in the frequency range of interest (up [illegible] 1000 MHz) compared to the inherent errors of the typical data-taking apparatus (where 5 percent errors are considered reasonable).

Also plots are includ[illegible] where inverse transforms applied to the (analytic) frequency spectra of given functions, two functions are compared to the original functions used:

a. Mimipulse (fig. 15-20)

b. Double exponential, i.e., $e^{\alpha t} - e^{\beta t}$ (fig. 21-23).

One might conclude from these results that FLIT performs quite credibly. However, the whole area of inverse Fourier transforms is fraught with danger. In the event that the time frequency interval or both do not cover the (significant) domain of the function, results from (1) the FFT, (2) the FFT with the value at 0 subtracted, or (3) FLIT can be unreliable. Cases can easily be constructed by truncating the time or frequency range of an analytic function—where either (1), (2), or (3) will give the best results. Since all three converge (pointwise) to the correct value as the frequency range and N go to infinity, one may increase the number of points in the array, provided the data and computation facilities support this increase. Lack of convergence is often indicated by failure of FLIT to return to 0 or of the FFT beginning significantly far from 0. Convergence is often indicated when the FFT and FLIT agree. When the data support it, INUFT, with its ability to accept nonequispaced frequency records, may be the transform chosen.

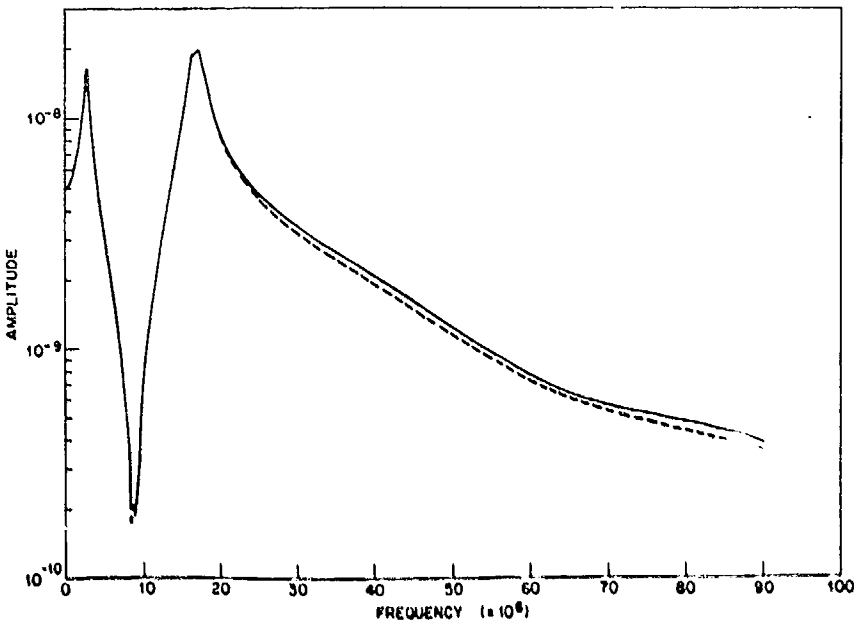

Figure 1. Absolute values of analytic transform of Mimipulse (solid line) and FLAT of equispaced Mimipulse for 256 points (dotted line).

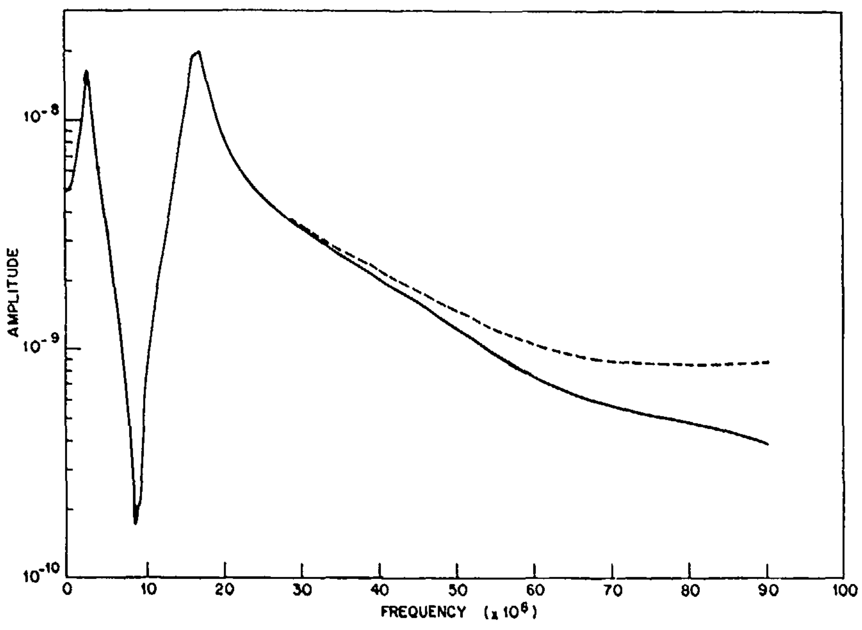

Figure 2. Absolute values of analytic transform of Mimipulse (solid line) and FFT of equispaced Mimipulse for 256 points (dotted line).

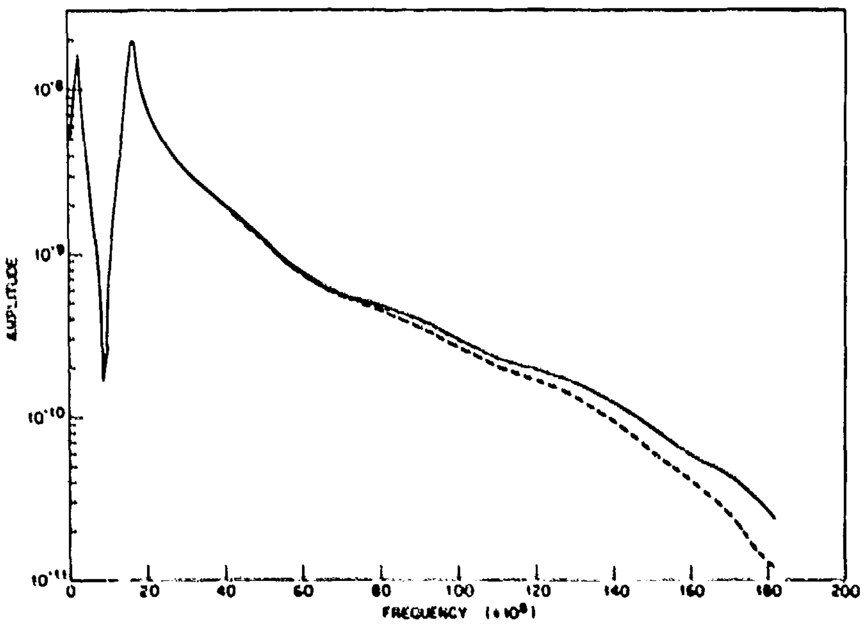

Figure 3. Absolute values of analytic transform of Mimipulse (solid line) and FLAT of equispaced Mimipulse for 512 points (dotted line).

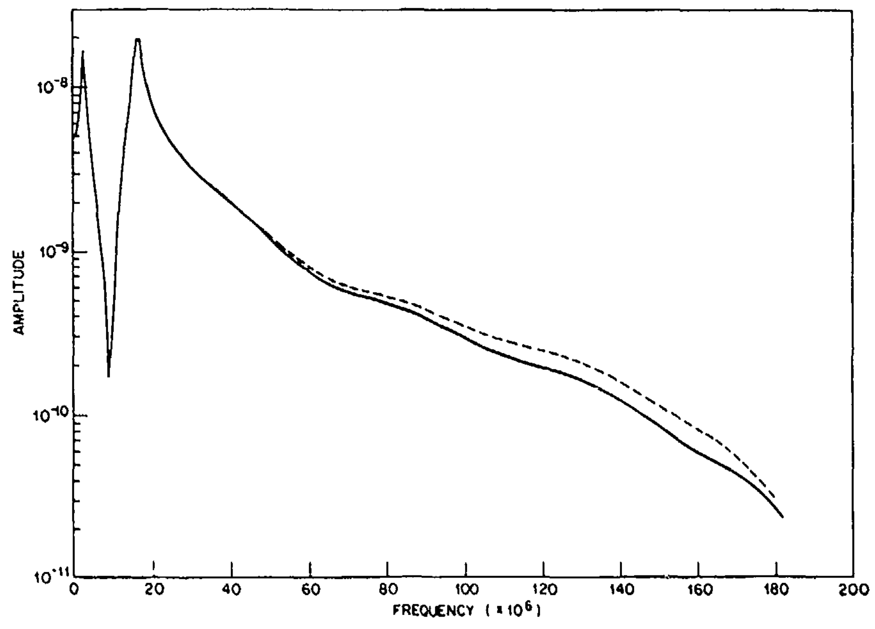

Figure 4. Absolute values of analytic transform of Mimipulse (solid line) and FFT of equispaced Mimipulse for 512 points (dotted line).

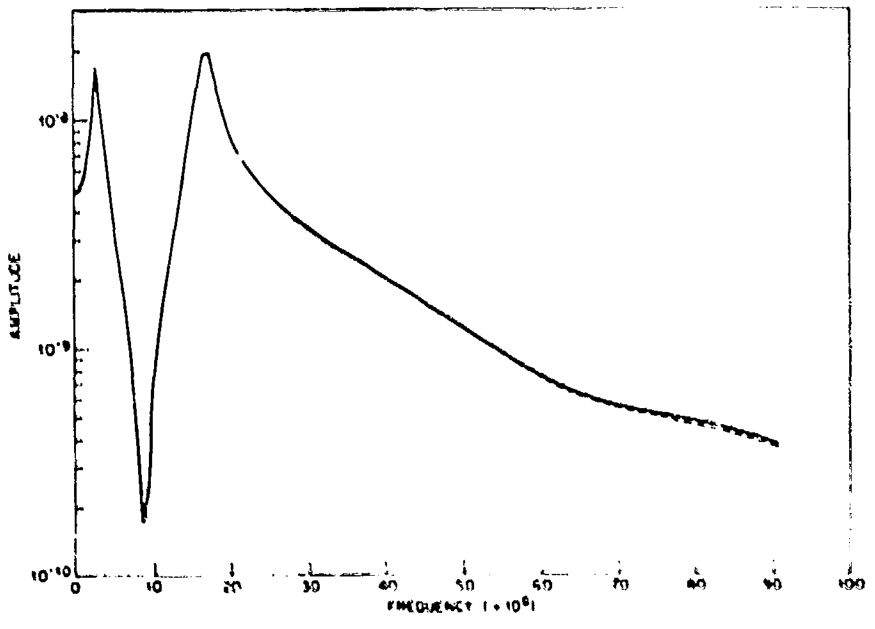

Figure 5. Absolute values of analytic transform of Mimipulse (solid line) and FLAT (first 90 MHz) of equispaced Mimipulse for 1024 points (dotted line).

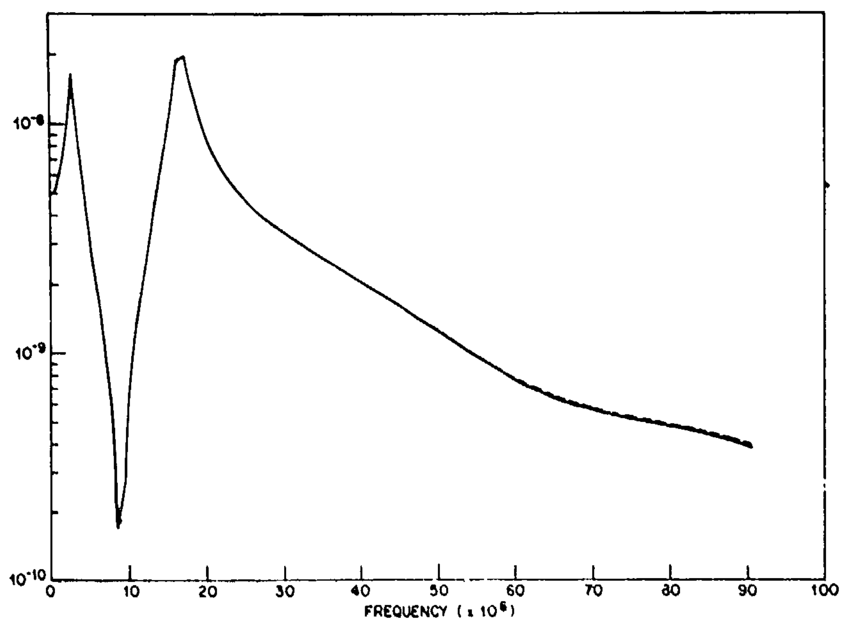

Figure 6. Absolute values of analytic transform of Mimipulse (solid line) and FFT (first 90 MHz) of equispaced Mimipulse for 1024 points (dotted line).

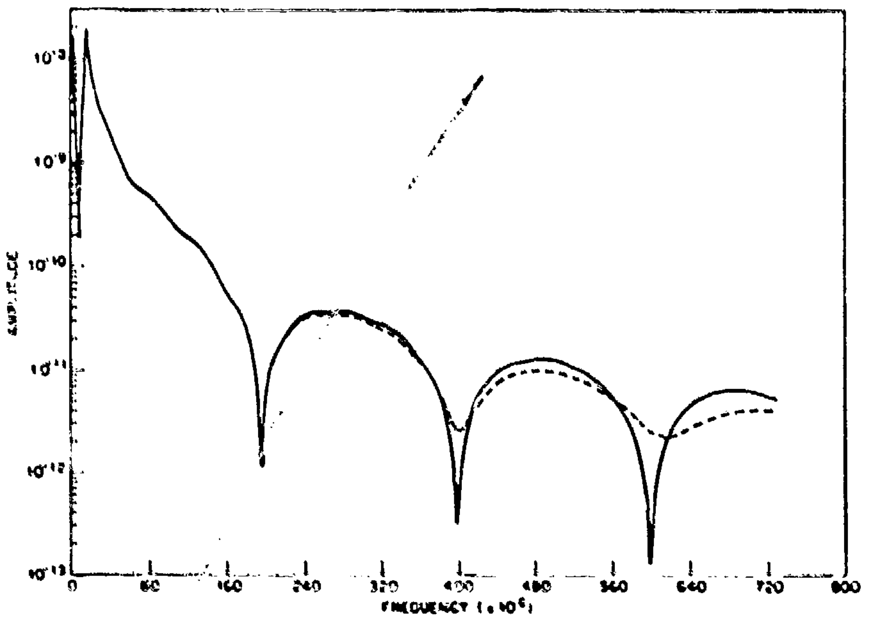

Figure 7. Absolute values of analytic transform of Mimipulse (solid line) and FLAT of equispaced Mimipulse for 2048 points (dotted line).

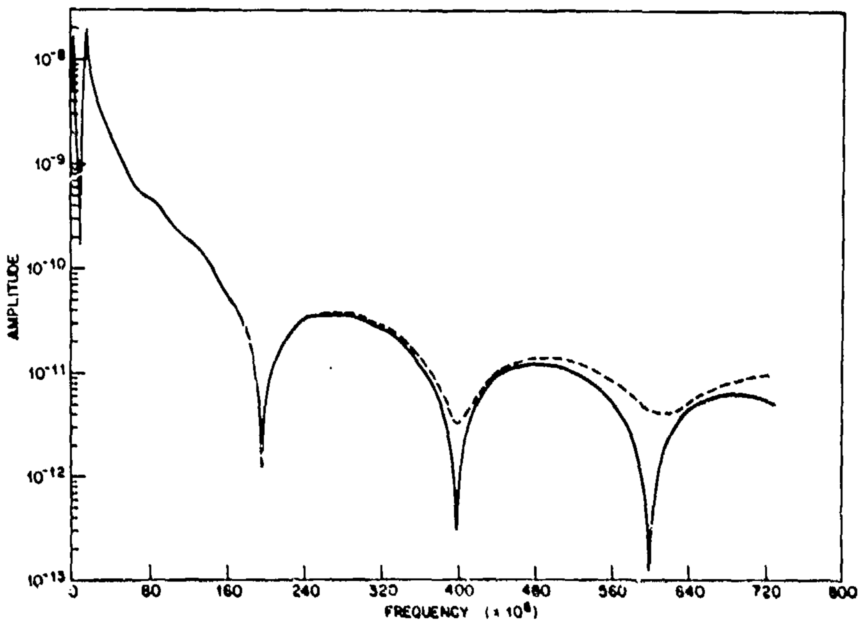

Figure 8. Absolute values of analytic transform of Minipulse (solid line) and FFT of equispaced Minipulse for 2048 points (dotted line).

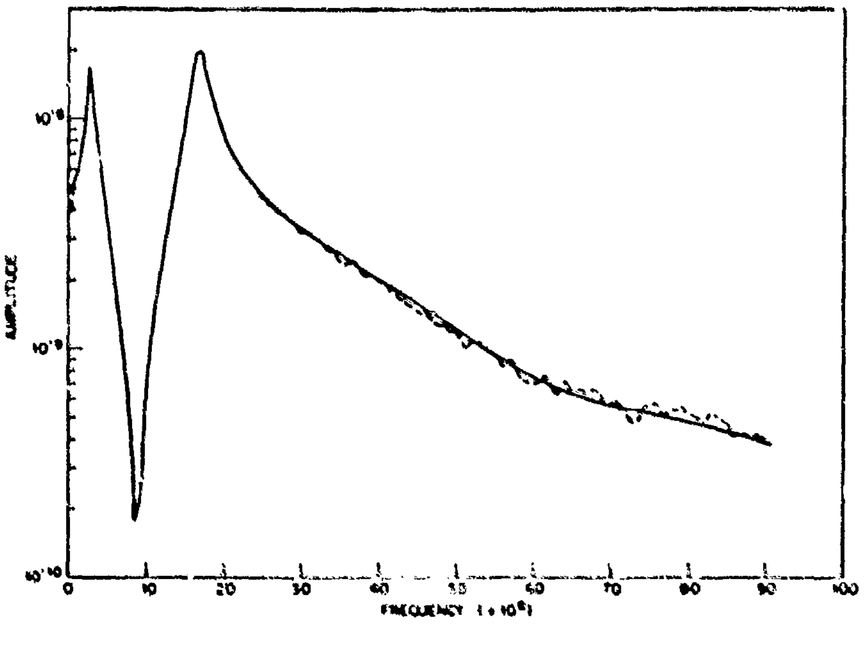

Figure 9. First 90 MHz of analytic transform of Minipulse (solid line) and FLAT of digitized Minipulse 2048 points interpolated array (dotted line).

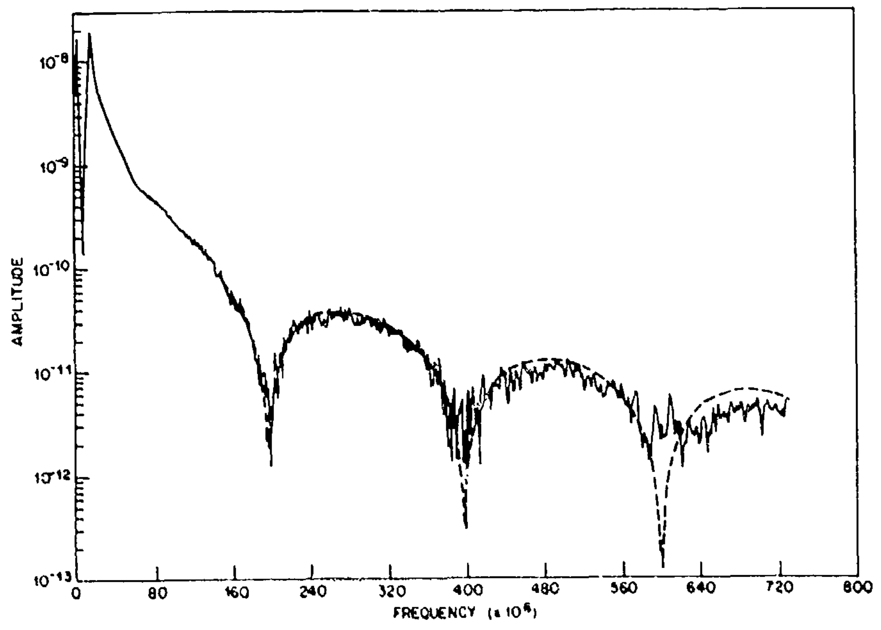

Figure 10. Analytic transform of Mimipulse (dotted line) and FLAT of digitized Mimipulse 2048 point interpolated array (solid line).

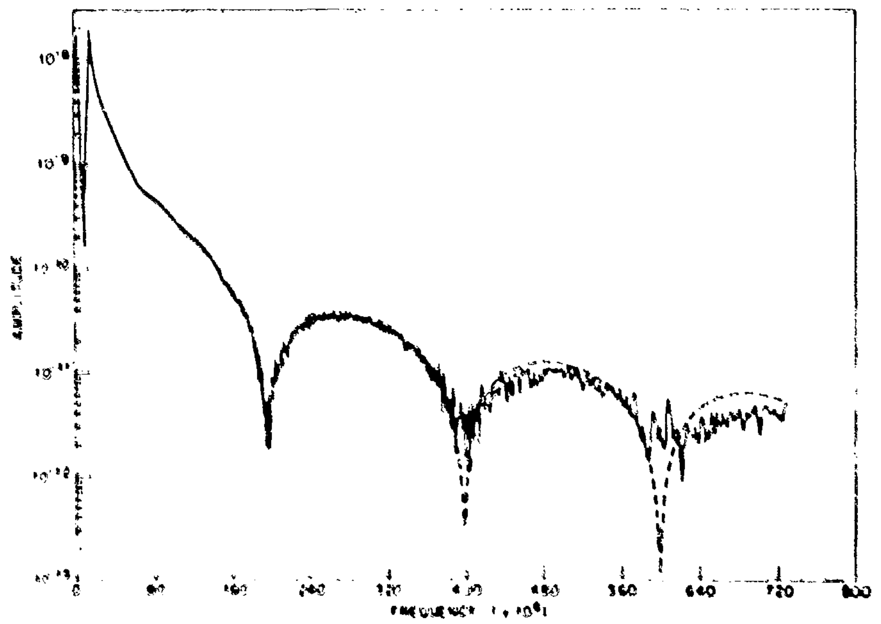

Figure 11. Analytic transform of Mimipulse (dotted line) and FLAT of array whose values are analytic values of Mimipulse at digitized time points interpolated to 2048 points to simulate sampling errors (solid line).

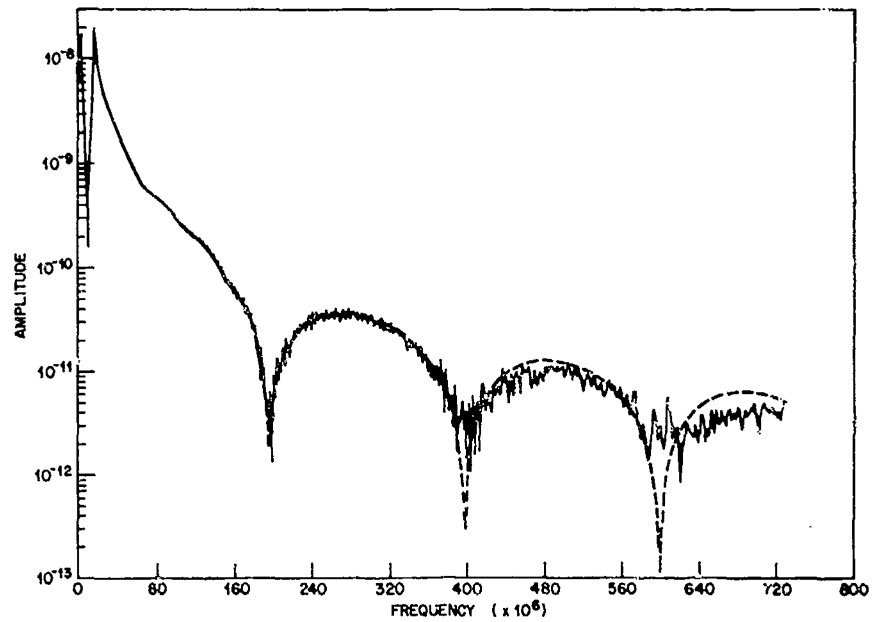

Figure 12. FLAT of analytic values of Mimipulse evaluated at digitized time points, quantified to multiples of 1/1000 of maximum value and linearly interpolated to 2048 points, simulating sampling and quantification errors (solid line) and analytic transform of Mimipulse (dotted line).

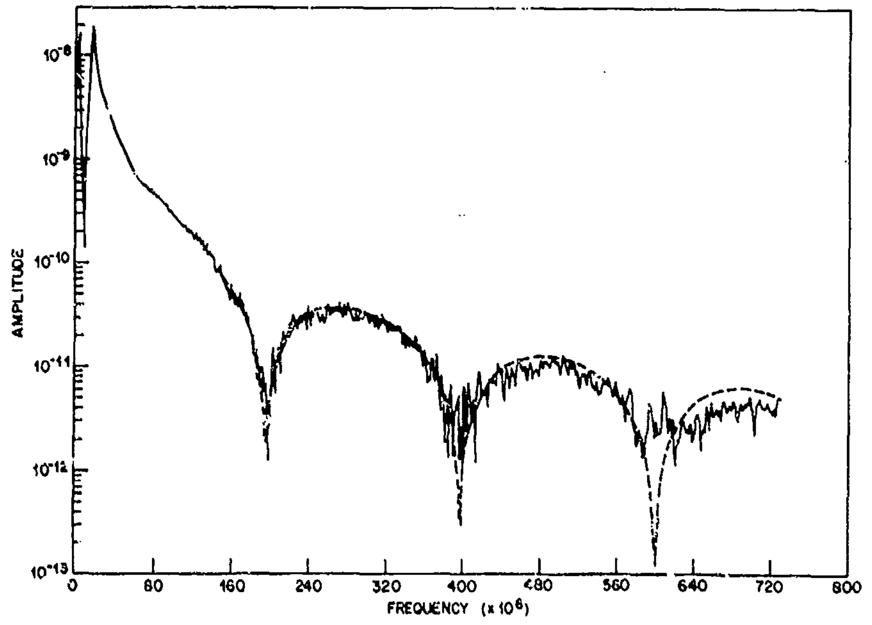

Figure 13. FLAT of analytic values of Mimipulse evaluated at digitized time points quantified to multiples of 1/1000 of maximum value and R multiples of 1/1000 of maximum value, where R is random number between -10 and 10, simulating effects of digitization (solid line) and analytic transform of Mimipulse (dotted line).

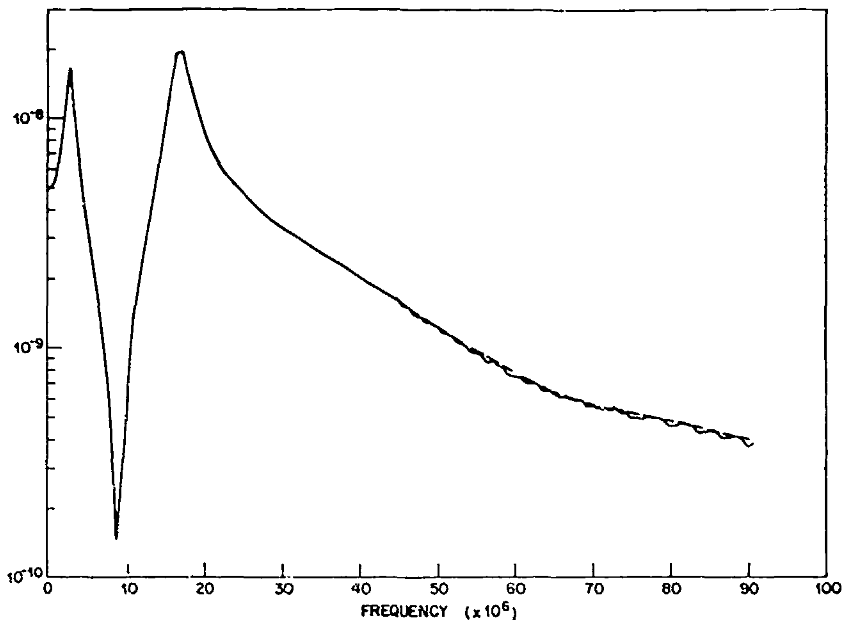

Figure 14. First 90 MHz of graph in figure 13.

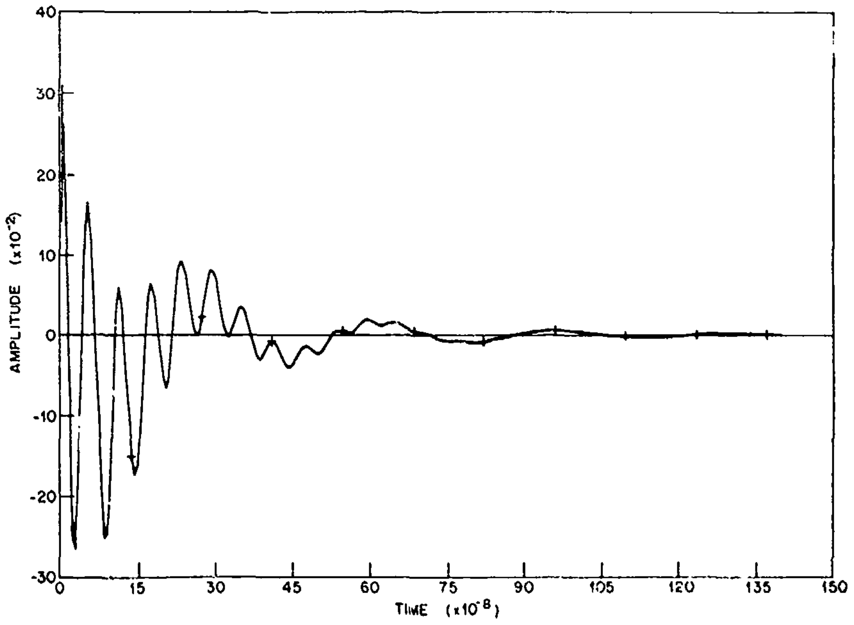

Figure 15. Mimipulse (solid line) and FLIT of analytic transform of Mimipulse for 256 points (crossed line).

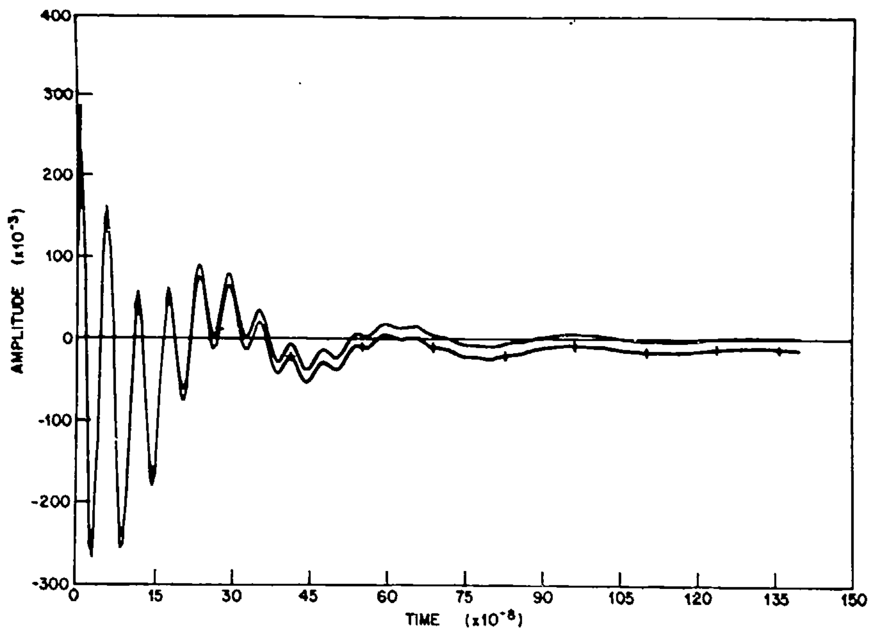

Figure 16. Mimipulse (solid line) and double aliased FFT of analytic transform of Mimipulse for 256 points, with value at 0 subtracted (crossed line).

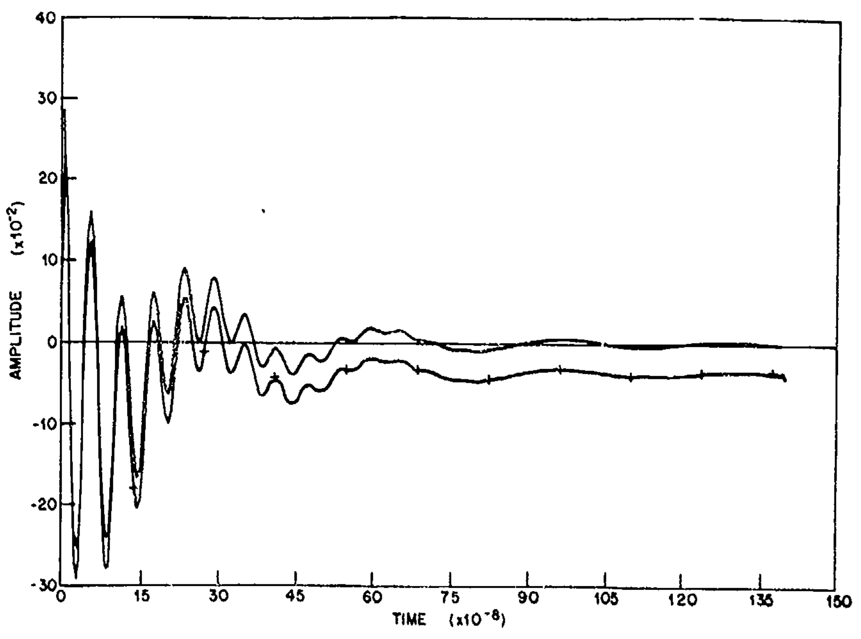

Figure 17. Mimipulse (solid line) and single aliased FFT of the analytic transform of Mimipulse for 256 points, with value at 0 subtracted (crossed line).

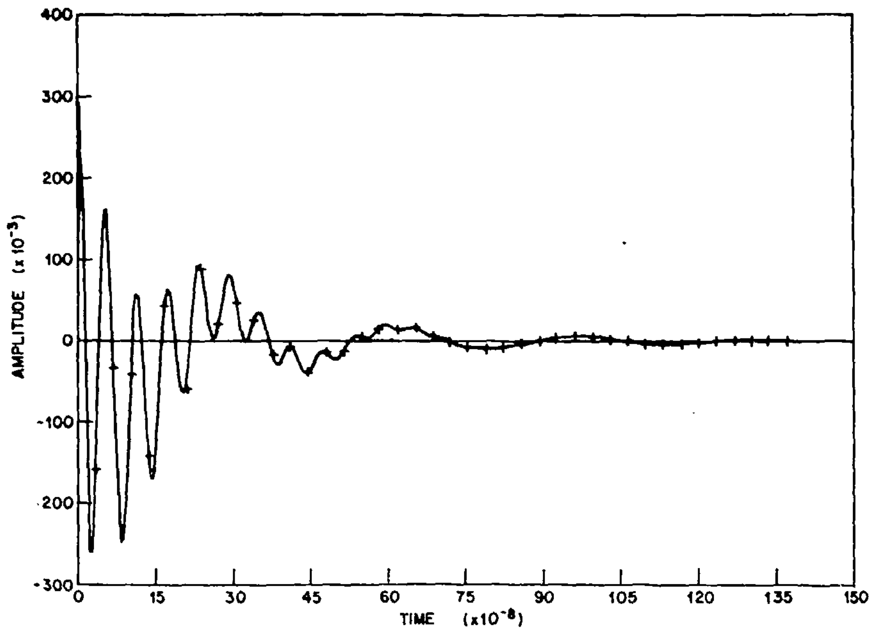

Figure 18. Mimipulse (solid line) and FLIT of analytic transform of Mimipulse for 1024 points (crossed line).

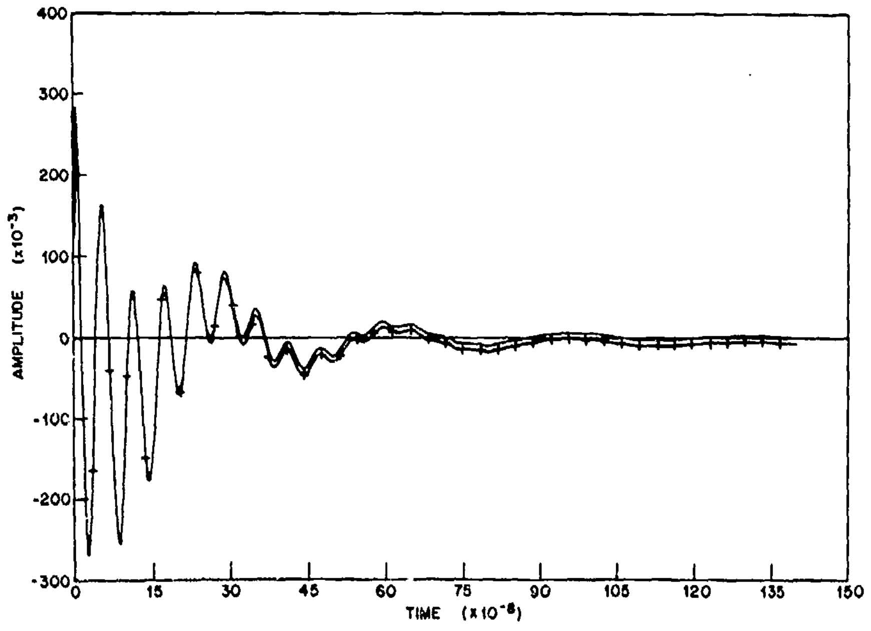

Figure 19. Mimipulse (solid line) and double aliased FFT of analytic transform of Mimipulse for 1024 points (crossed line).

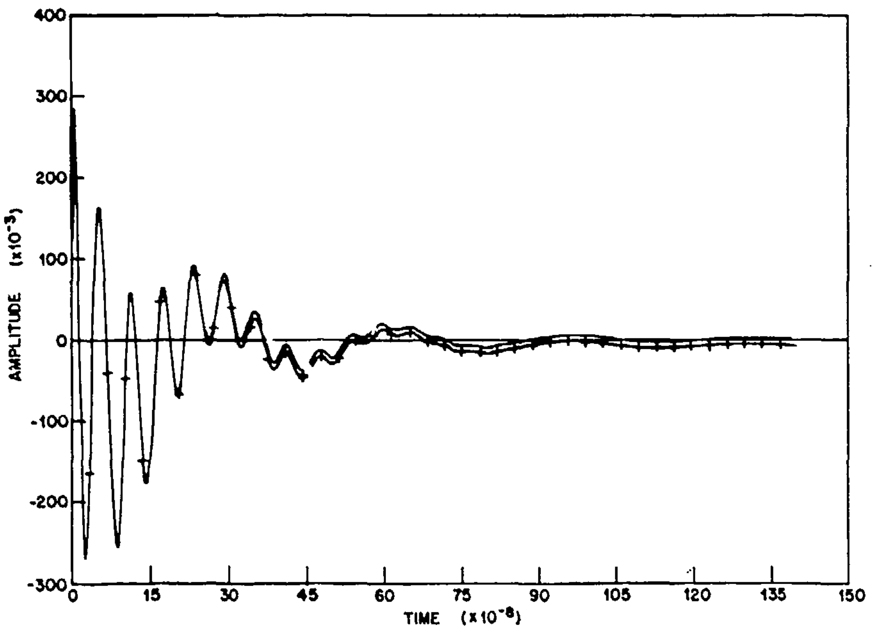

Figure 20. Mimipulse (solid line) and single aliased FFT of analytic transform of Mimipulse for 1024 points (crossed line).

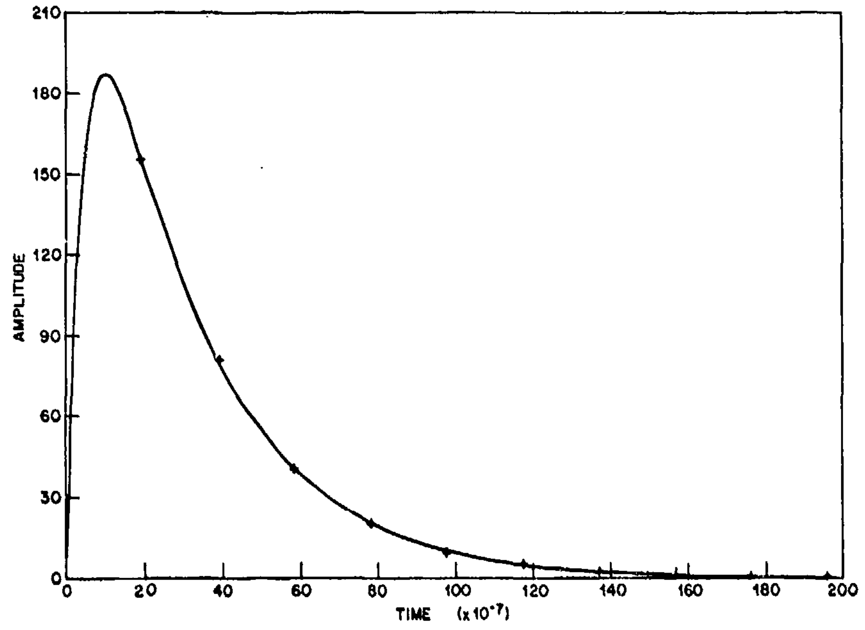

Figure 21. FLIT of analytic transform of $k\left(e^{\alpha t} - e^{\beta t}\right)$ for 256 points (solid line) and $k\left(e^{\alpha t} - e^{\beta t}\right)$ (crossed line).

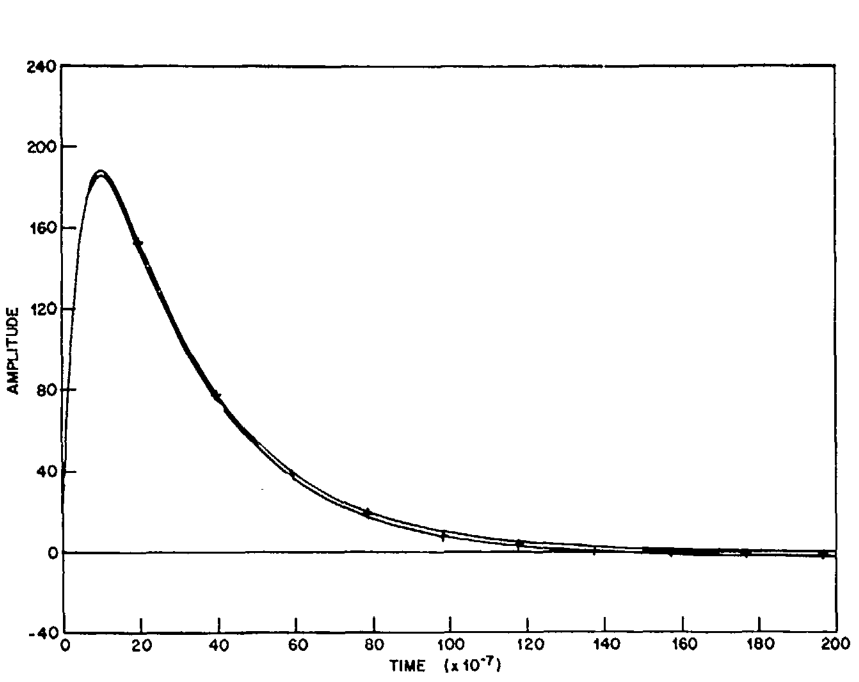

Figure 22. Graph of $k\left(e^{at} - e^{\beta t}\right)$ (solid line) and double aliased FFT of analytic transform of $k\left(e^{at} - e^{\beta t}\right)$ for 256 points with value at 0 substracted (crossed line).

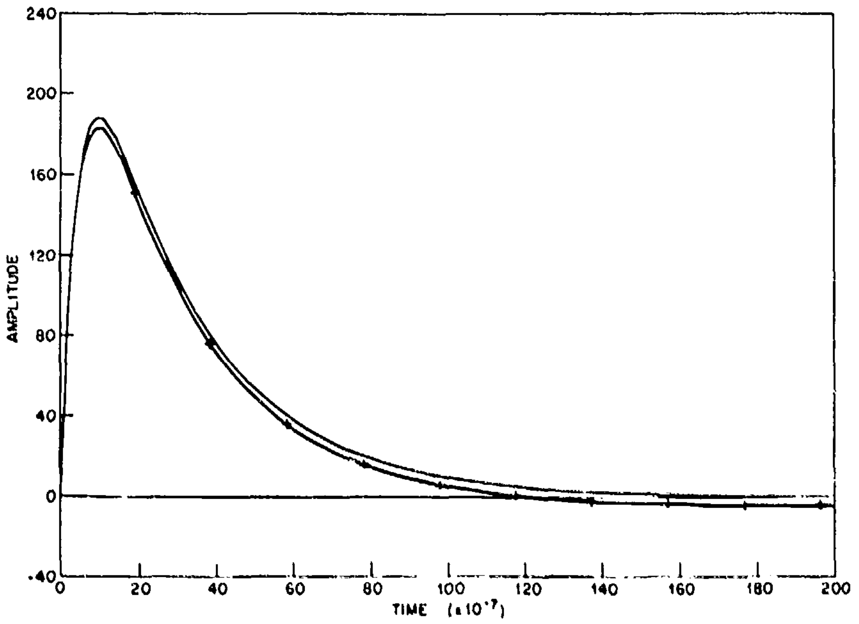

Figure 23. Graph of $k\left(e^{at} - e^{\beta t}\right)$ (solid line) and single aliased FFT of analytic transform $k\left(e^{at} - e^{\beta t}\right)$ with value at 0 subtracted (crossed line).

10. SUBROUTINES RDTAPE AND CSTOUT

Subroutines RDTAPE and CSTOUT were written by T.V. Noon of HDL, and they are used to read the digitized data for storage and prepare them for input for further analysis. Subroutine RDTAPE reads (possibly) multiple data sets consisting of binary information, fills up the appropriate arrays, and checks for the end of the collection of data and for irregularities in the data format.

Subroutine RDTAPE may be used in two modes, only one of which is of interest here. It is called by

```
Call RDTAPE(NT, X,Y,N,LABEL)
```

NT	=	number of file containing data (integer)
X	=	name of array to receive independent variable,
Y	=	name of array to receive dependent variable,
N	=	name of (integer) variable to receive number of points,
LABEL	=	name of array to receive label (eight words),

Subroutine CSTOUT checks the output of RDTAPE for monotonicity. If

$$X(I+1) = X(I+2) = \ldots = X(I+n),$$

then

$$X(I+2), \ldots, X(I+n), Y(I+2), \ldots, Y(I+n)$$

are removed from the respective arrays, the arrays are reordered, and Y(I + 1) is replaced by

$$\sum_{i=1}^{n} Y(I+i)/n.$$

If

$$X(I+2) < X(I+1) ,$$

then

$$X(I+2), Y(I+2)$$

are removed from the respective arrays, and the arrays are reordered. The routine is called by

```
Call CSTOUT(X,Y,N)
```

where

X	=	name of independent array,
Y	=	name of dependent array,
N	=	number of points--possibly a new value will be returned.

APPENDIX A.–LISTINGS OF PROGRAMS USED IN NUMERICAL FOURIER TRANSFORMS

```
      SUBROUTINE FLAT(YNYQA,NSTAR,DT,JFLAG)
      COMPLEX X,YNYQA(NSTAR),YV1,YV2,XM,TCI,A
C FLAT CALCULATES THE TRANSFORM TO FREQUENCY SPACE OF THE COMPLEX ARRAY
C GIVEN IN YNYQA. NSTAR IS THE NUMBER OF POINTS IN THE ARRAY.. DT IS
C DELTA TIME FOR THE ARRAY. JFLAG SHOULD BE SET TO -1  IF THE TRANSFORM
C IS TO BE EXPRESSED IN TERMS OF EXP(-2*PI*F),+1 OTHERWISE. THE TRANSFORM
C IS DONE IN PLACE.
      YV1=0.0 $ NPT=NSTAR-1 $ SNYQ=0.0
      TPI=8.*ATAN(1.)        $ TPI2I=(-1./(TPI*TPI))
      TMAX=NPT*DT
      N1=NSTAR/2+1  $ N2=N1-2
      DF=1./(NSTAR*DT)
      TCI=JFLAG*DF*CMPLX(0.,TPI)
      T=DT*NPT
      A=YNYQA(NSTAR)
      DO 10 I=2,NPT
      SNYQ=SNYQ+REAL(YNYQA(I))
10    CONTINUE
      SNYQ=SNYQ+.5*REAL(YNYQA(NSTAR))
600   DO 110 I=1,NPT
      YV1=YNYQA(I)
      YV2=YNYQA(I+1)
110   YNYQA(I)=YV2-YV1
      YNYQA(NSTAR)=-YV2
      YV2=0.
500   DO 115 I=1,NSTAR
      YV1=YNYQA(I)
      YNYQA(I)=(YV2-YV1)/DT
115   YV2=YV1
      YNYQA(1)=YNYQA(1)+YV1/DT
      CALL FFT1D(YNYQA,NSTAR,JFLAG)
      YNYQA(1)=SNYQ*DT
      DO 175 I=2,N1
      X=(TCI*(I-1))**2
175   YNYQA(I)=-YNYQA(I)/X
      DO 176 I=1,N2
176   YNYQA(N1+I)=CONJG(YNYQA(N1-I))
      RETURN
      END

      SUBROUTINE FLIT (YNYQA,NSTAR,DF,JFLAG)
      COMPLEX YNYQA(NSTAR),W,S
C FLIT CALCULATE THE TRANSFORM TO TIME SPACE OF THE COMPLEX ARRAY GIVEN
C IN YNYQA. NSTAR IS THE NUMBER OF POINTS. DF IS THE FREQUENCY INCREMENT
C JFLAG SHOULD BE SET TO +1 IF THE FREQUENCY ARRAY IS EXPRESSED IN TERMS
C OF EXP(-2*PI*F)., -1 OTHERWISE. OUTPUT IS A COMPLEX ARRAY YNYQA
C WHOSE REAL PART IS THE DESIRED TRANSFORM.
      A=REAL(YNYQA(1))
      DT=1./(DF*NSTAR)
      TPI=8.*ATAN(1.)
      N1=NSTAR/2+1
      N2=N1-2
      FAC=-(TPI*DF)**2/NSTAR
      DO 105 I=1,N1
105   YNYQA(I)=YNYQA(I)*FAC*(I-1)**2
      DO 110 I=1,N2
      YNYQA(N1+I)=CONJG(YNYQA(N1-I))
```

Preceding page blank

```
 110  CONTINUE
      CALL FFT1D(YNYQA,NSTAR,JFLAG)
      XL=0.
      DO 75 I=2,NSTAR
      XL=XL+REAL(YNYQA(I))
75    CONTINUE
      YNYQA(1)=-XL
      W=0.  $ T=0.
      S=0.
      DO 50 I=1,NSTAR
      S=S+YNYQA(I)
      YNYQA(I)=W
50    W=W+S*DT
      T2=0.
      T=DT*(NSTAR-1)
      X=0.
      DO 325 I=1,NSTAR
325   X=X+REAL(YNYQA(I))
      N=NSTAR
      X2=(A/DT-X)*2./(N*(N-1))
      DO 435 I=1,NSTAR
435   YNYQA(I)=YNYQA(I)+(I-1)*X2
      RETURN
      END
      SUBROUTINE ANMI(X,Y,N)
      DIMENSION X(N),Y(N)
      NAMELIST /PULSE/ A,AA,T1,T2,T3,T4,TE,S1,S2,S3,S4
      PI=4.*ATAN(1.)
      READ PULSE
      PRINT PULSE
      W1=PI/(T4-T3)/2.
      W2=PI*S4/(TE-T3)
      X1=1./S1/(T2-T1)
      X2=1./S2/(TE-T3)
      X3=1./S3/(TE-T3)
      YY=-PI*A/2./(T2-T1)*EXP (-X1*(T2-T1))
      GG=-2.*AA/(T3-T2)**3+YY/(T3-T2)**2
      DG=-AA/(T3-T2)**2-GG*(T2+2.*T3)
      C1=A/T1
      C21=T2-T1
      C22=PI/2./C21
      C44=X2/W2
      DO 16 I=1,N
      IF(X(I).GT.T1) GO TO 17
16    CONTINUE
17    N1=I-1
      DO 18 I=N1,N
      IF(X(I).GT.T2) GO TO 19
18    CONTINUE
19    N2=I-1
      DO 20 I=N2,N
      IF(X(I).GT.T3) GO TO 21
20    CONTINUE
21    N3=I-1
      DO 22 I=N3,N
      IF(X(I).GT.TE) GO TO 23
22    CONTINUE
      N4=N
      GO TO 24
23    N4=I-1
24    DO 30 I=1,N1
30    Y(I)=C1*X(I)
      CC21=A*EXP(X1*T1)
      N1=N1+1
      DO 40 I=N1,N2
      TI=X(I)
40    Y(I)=CC21*EXP(-X1*TI)*COS(C22*(TI-T1))
      N2=N2+1
      BG=(-2.*DG-3.*GG*T3)*T3
      AG=(-BG-(DG+GG*T2)*T2)*T2
      DO 50 I=N2,N3
      TI=X(I)
50    Y(I)=AG+(BG+(DG+GG*TI)*TI)*TI
      N3=N3+1
      CC41=AA*EXP(X2*T3)
      CC46=W1*T3
      CC45=AA*C44*EXP(X3*T3)
      CC48=W2*T3
      DO 60 I=N3,N
      TI=X(I)
60    Y(I)=CC41*EXP(-X2*TI)*COS(W1*TI-CC46)+CC45*EXP(-X3*TI)
```

```
     1 *SIN(W2*TI-CC48)
       IF(N4.EQ.N) RETURN
       DO 70 I=N4,N
70     Y(I)=0.
       RETURN
       END

       SUBROUTINE ANTRA(OM,FT,NPTS)
       COMPLEX FT(NPTS),FTI,CWT1,CWT3,CWT4,CM2,CM3,EYE
       COMPLEX CWT,CWT2,FT1,FT2,FT3,FT4,CM1
       DIMENSION OM(NPTS)
       NAMELIST /PULSE/ A,AA,T1,T2,T3,T4,TE,S1,S2,S3,S4,FINIT
       PI=4.*ATAN(1.)
       EYE=CMPLX(0.,1.)
       READ PULSE
       PRINT PULSE
       W1=PI/(T4-T3)/2.
       W2=PI*S4/(TE-T3)
       X1=1./S1/(T2-T1)
       X2=1./S2/(TE-T3)
       X3=1./S3/(TE-T3)
       Y=-PI*A/2./(T2-T1)*EXP (-X1*(T2-T1))
       GG=-2.*AA/(T3-T2)**3+Y/(T3-T2)**2
       DG=-AA/(T3-T2)**2-GG*(T2+2.*T3)
       BG=(-2.*DG-3.*GG*T3)*T3
       AG=(-BG-(DG+GG*T2)*T2)*T2
       C01=T3-T2
       C1=A/T1
       C21=T2-T1
       C22=PI/2./C21
       C23=C22**2
       C24=EXP(-X1*C21)
       C31=2.*DG+6.*GG*T3
       C32=2.*DG+6.*GG*T2
       C33=6.*GG
       C41=TE-T3
       C42=W1**2
       C43=W2**2
       C44=X2/W2
       C45=COS(W1*C41)
       C46=SIN(W1*C41)*W1
       C47=SIN(W2*C41)
       C48=COS(W2*C41)*W2
       C49=EXP(-X2*C41)
       C410=EXP(-X3*C41)*C44
       DO 100 I=1,NPTS
       W=OM(I)
       WT=W*T1
       WS=W**2
       WQ=WS*W
       WF=WQ*W
       WT1=W*C21
       WT2=W*T2
       WT3=W*T3
       WT4=W*C41
       CWT=CMPLX(COS(WT),-SIN(WT))
       CWT1=CMPLX(COS(WT1),-SIN(WT1))
```

```
      CWT2=CMPLX(COS(WT2),-SIN(WT2))
      CWT3=CMPLX(COS(WT3),-SIN(WT3))
      CWT4=CMPLX(COS(WT4),-SIN(WT4))
      CM1=CMPLX(X1,W)
      CM2=CMPLX(X2,W)
      CM3=CMPLX(X3,W)
      IF(W.NE.0.0) GO TO 92
      FTI    =.5*A*T1
      FTI=FTI+(AG+BG*(T3+T2)*.5+DG*(T3**2+T3*T2+T2**2)/3.+GG*
     1 (T3**3+T3**2*T2+T3*T2**2+T2**3)*.25)*C01
      GO TO 94
92    FTI=CWT *CMPLX(1.,WT)-1
      FTI    =C1*FTI/WS
      FT1=AA*EYE/W-EYE*C31/WQ-C33/WF
      FT2=Y/WS-EYE*C32/WQ-C33/WF
      FTI=FTI+CWT3*FT1-CWT2*FT2
94    FT1=CM1*CM1+C23
      FT2=C24*CWT1 *C22+CM1
      FT1=A*CWT /FT1
      FTI=FTI+FT1*FT2
      FT1=AA*CWT3
      FT2=-CM2*C45+C46
      FT2=C49*CWT4*FT2*FINIT+CM2
      FT3=-CM3*C47-C48
      FT3=C410*CWT4*FT3*FINIT+C44*W2
      FT4=CM2*CM2+C42
      FT2=FT2/FT4
      FT4=CM3*CM3+C43
      FT3=FT3/FT4
100   FT(I)=FTI+FT1*(FT2+FT3)
      RETURN
      END
```

APPENDIX A

```
      SUBROUTINE NUFT(X,Y,N,NU,OM,IOM,FT,DX,JFLAG)
C THIS SUBROUTINE CALCULATES THE FOURIER  TRANSFORM OF THE
C FUNCTION GIVEN BY STRAIGHT LINES JOINING THE POINTS GIVEN BY THE
C ARRAYS X,Y. THE NUMBER OF INPUT POINTS IS N, THE FREQUENCY ARRAY OM IS
C PROVIDED AND HAS DIMENSION NU. DX IS A REAL ARRAY OF DIMENSION N
C THAT IS USED FOR SCRATCH. THE FOURIER TRANSFOR IS GIVEN IN THE
C COMPLEX ARRAY FT.
C IF IOM=1, INPUT AND OUTPUT OM ARE CIRCULAR FREQUENCIES
C IF IOM=2,INPUT ARRAY IS CIRCULAR FREQUENCY, OUTPUT IS FREQUENCY
C IF IOM=3, INPUT ARRAY IS FREQUENCY, OUTPUT IS CIRCULAR FREQUENCY
C IF IOM=4, INPUT AND OUTPUT OM ARE FREQUENCIES
C JFLAG=+1 IF THE FOURIER TRANSFORM HAS A FACTOR EXP(+IWT)
C JFLAG=-1 IF THE FOURIER TRANSFORM HAS A FACTOR EXP(-IWT)
      DIMENSION X(N),Y(N),DX(N),OM(NU)
      COMPLEX EYE,EX1,EX2,S1,S2,FT(NU)
      TOPI=8.*ATAN(1.)
      TPIN=1./TOPI
      S=0.
      EYE=(0.,1.)
      IF(IOM.LE.2) GO TO 25
      DO 20 I=1,NU
20    OM(I)=OM(I)*TOPI
25    NP=N-1
      DO 30 I=1,NP
30    DX(I)=X(I+1)-X(I)
      NO=1
      IF(OM(1).NE.0.) GO TO 42
      NO=2
      DO 40 I=1,NP
40    S=S+(Y(I+1)+Y(I))*DX(I)
      FT(1)=CMPLX(S/2,0.0)
42    DO 45 I=1,NP
45    DX(I)=(Y(I+1)-Y(I))/DX(I)
      DO 70 I=NO,NU
      W=OM(I)
      WS=W*W
      WC=WS*W
      WF=WC*W
      S2=(0.0,0.0)
      IFLAG=0
      WT=W*X(1)
      EX1=CMPLX(COS(WT),SIN(WT))
      S1=Y(1)*EX1
      DO 50 J=2,N
      IF (IFLAG.EQ.1) GO TO 47
      X2=X(J)
      X1=X(J-1)
      IF((W*X2).GT.5.0E-2) GO TO 46
      XP=X1+X2
      XQ=XP*X1+X2**2
      XR=XQ*X1+X2**3
      DY=Y(J)-Y(J-1)
      XRE=(-0.5*WS*XP+WF*XR/24.)*DY
      XIM=(W-WC*XQ/6.)*DY
      S2=S2+CMPLX(XRE,XIM)
      GO TO 50
46    EX1=CMPLX(COS(W*X1),SIN(W*X1))
```

```
      IFLAG=1
47    WT=W*X(J)
      EX2=CMPLX(COS(WT),SIN(WT))
      S2=S2+(EX2-EX1)*DX(J-1)
      EX1=EX2
50    CONTINUE
      IF(IFLAG.EQ.1) GO TO 48
      EX1=CMPLX(COS(WT),SIN(WT))
48    S1=S1-Y(N)*EX1
70    FT(I)=(EYE*S1+S2/W)/W
      IF(IOM.EQ.1.OR.IOM.EQ.3) GO TO 85
      DO 80 I=1,NU
80    OM(I)=OM(I)*TPIN
85    IF(JFLAG.EQ.1) RETURN
      DO 90 I=1,NU
90    FT(I)=CONJG(FT(I))
      RETURN
      END

      SUBROUTINE INVFT(X,Y,N,NU,OM,IOM,FT,CSCR,JFLAG)
C THIS SUBROUTINE CALCULATES THE INVERSE FOURIER TRANSFORM OF THE FUNCTION
C GIVEN BY STRAIGHT LINES JOINING THE POINTS GIVEN BY THE COMPLEX ARRAY FT
C AS A FUNCTION OF THE REAL ARRAY OM. THE NUMBER OF INPUT POINTS IS NU.
C THE INVERSE F.T. IS ASSUMED TO BE A REAL FUNCTION AND GOES IN THE
C ARRAY Y, AND THE REAL ARRAY X IS GIVEN AND HAS DIMENSION N.
CSCR IS A COMPLEX ARRAY OF DIMENSION NU AND IS USED FOR SCRATCH
C IF IOM=1, INPUT AND OUTPUT OM ARE CIRCULAR FREQUENCIES
C IF IOM=2,INPUT ARRAY IS CIRCULAR FREQUENCY, OUTPUT IS FREQUENCY
C IF IOM=3, INPUT ARRAY IS FREQUENCY, OUTPUT IS CIRCULAR FREQUENCY
C IF IOM=4, INPUT AND OUTPUT OM ARE FREQUENCIES
C JFLAG=+1 IF THE INVERSE FOURIER TRANSFORM HAS A FACTOR EXP(+IWT)
C JFLAG=-1 IF THE INVERSE FOURIER TRANSFORM HAS A FACTOR EXP(-IWT)
      DIMENSION X(N),Y(N),OM(NU)
      COMPLEX EYE,EX1,EX2,S1,S2,FT(NU),CSCR(NU),S,OV,WRE,WIM
      TOPI=8.*ATAN(1.)
      TPIN=1./TOPI
      PIN=2.*TPIN
      S=0.
      EYE=CMPLX(0.,1.)
      IF(JFLAG.EQ.1) GO TO 15
      DO 10 I=1,NU
10    FT(I)=CONJG(FT(I))
15    IF(IOM.LE.2) GO TO 25
      DO 20 I=1,NU
20    OM(I)=OM(I)*TOPI
25    NP=NU-1
      NO=1
      IF( X(1).NE.0.) GO TO 42
      NO=2
      DO 40 I=1,NP
40    S=S+(FT(I+1)+FT(I))*(OM(I+1)-OM(I))
      Y(1)=REAL(S)*TPIN
42    DO 45 I=1,NP
45    CSCR(I)=(FT(I+1)-FT(I))/(OM(I+1)-OM(I))
      DO 70 I=NO,N
      W=X(I)
      WS=W*W
      WC=WS*W
      WF=WC*W
      S2=(0.0,0.0)
      IFLAG=0
      WT=W*OM(1)
      EX1=CMPLX(COS(WT),SIN(WT))
      S1=FT(1)*EX1
      DO 50 J=2,NU
      IF (IFLAG.EQ.1) GO TO 47
      X2=OM(J)
      X1=OM(J-1)
      IF((W*X2).GT.5.0E-2) GO TO 46
      XP=X1+X2
      XQ=XP*X1+X2**2
      XR=XQ*X1+X2**3
      DY=FT(J)-FT(J-1)
      XRE=(-0.5*WS*XP+WF*XR/24.)*DY
      XIM=(W-WC*XQ/6.)*DY*EYE
      S2=S2+XRE+XIM
      GO TO 50
46    EX1=CMPLX(COS(W*X1),SIN(W*X1))
      IFLAG=1
47    WT=W*OM(J)
      EX2=CMPLX(COS(WT),SIN(WT))
      S2=S2+(EX2-EX1)*CSCR(J-1)
      EX1=EX2
50    CONTINUE
      IF(IFLAG.EQ.1) GO TO 48
      EX1=CMPLX(COS(WT),SIN(WT))
48    S1=S1-FT(NU)*EX1
      S     =(EYE*S1+S2/W)/W
      Y(I)=REAL(S)*PIN
70    CONTINUE
      IF(IOM.EQ.1.OR.IOM.EQ.3) GO TO 85
      DO 80 I=1,NJ
80    OM(I)=OM(I)*TPIN
85    IF(JFLAG.EQ.1) RETURN
      DO  90 I=1,NU
90    FT(I)=CONJG(FT(I))
      RETURN
      END
      SUBROUTINE CLINT(XF,YF,JF,NSTAR,XNYQ,YNYQ)
      DIMENSION XF(JF),YF(JF)
      COMPLEX YNYQ(NSTAR)
      I=1
      X1=XF(1)
      YNYQ(1)=YF(1)
      DO 20 L=2,NSTAR
      X=X1+(L-1)*XNYQ
10    IF(X.LE.XF(I)) GO TO 20
      I=I+1
      IF(I.GT.JF) GO TO 30
      DENOM=XF(I)-XF(I-1)
      C1=(YF(I)-YF(I-1))/DENOM
      C2=(YF(I-1)*XF(I)-YF(I)*XF(I-1))/DENOM
      GO TO 10
20    YNYQ(L)=C1*X+C2
      GO TO 100
30    DO 40 J=L,NSTAR
40    YNYQ(J)=0.
100   RETURN
      END
      SUBROUTINE LINT(XF,YF,JF,NSTAR,XNYQ,YNYQ)
      DIMENSION XF(JF),YF(JF)
      DIMENSION YNYQ(NSTAR)
      I=1
      X1=XF(1)
      YNYQ(1)=YF(1)
      DO 20 L=2,NSTAR
      X=X1+(L-1)*XNYQ
10    IF(X.LE.XF(I)) GO TO 20
      I=I+1
      IF(I.GT.JF) GO TO 30
      DENOM=XF(I)-XF(I-1)
      C1=(YF(I)-YF(I-1))/DENOM
      C2=(YF(I-1)*XF(I)-YF(I)*XF(I-1))/DENOM
      GO TO 10
20    YNYQ(L)=C1*X+C2
      GO TO 100
30    DO 40 J=L,NSTAR
40    YNYQ(J)=0.
100   RETURN
      END
      SUBROUTINE RDTAPE(NT,X,Y,N,LABEL)
      DIMENSION X(1),Y(1),LABEL(8)
      READ(NT)(LABEL(I),I=1,8)
      IF(EOF(NT))40,10
   10 READ(NT)N
      IF(EOF(NT))50,20
   20 READ(NT)(X(I),Y(I),I=1,N)
      IF(EOF(NT))50,30
   30 RETURN
   40 PRINT 41
   41 FORMAT(5X,*EOF ENCOUNTERED PROPERLY*)
      GO TO 60
   50 PRINT 51,LABEL
   51 FORMAT(5X,*EOF ENCOUNTERED IMPROPERLY DURING *,8A10,* EXIT CALLED*
     *)
   60 CALL EXIT
      END

      SUBROUTINE CSTOUT(XF,YF,JF)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C     CHECK TIME ORDERING OF THE POINTS AND CAST OUT THOSE POINTS NOT IN THE  C
C     PROPER TIME  ORDER                                                 C
C     IT ALSO AVERAGES THOSE POINTS WHICH HAVE THE SAME X VALUES         C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION XF(JF),YF(JF)
      JBAD=0
      J=1
      LSUM=1
      K=2
      XSAV=XF(1)
      YSUM=YF(1)
80    IF(XF(K)-XSAV)90,100,110
90    K=K+1
      JBAD=JBAD + 1
      IF(K-JF)80,80,110
100   LSUM=LSUM+1
      YSUM=YSUM+YF(K)
      K=K+1
      IF(K-JF)80,80,30
110   YF(J)=YSUM/LSUM
      XF(J)=XSAV
      IF(K-JF)120,130,30
120   XSAV=XF(K)
      YSUM=YF(K)
      LSUM=1
      K=K+1
      J=J+1
      GO TO 80
130   J=J+1
      YF(J)=YF(K)
      XF(J)=XF(K)
   30 PRINT 1,JBAD,JF
1     FORMAT(/,1X,I5,* POINTS OUT OF A TOTAL OF *,I5,
     1 * DID NOT SATISFY THE TIME ORDER CRITERION*)
      JF=J
      JBAD=0
      I=1
201   IF(XF(I).GE.0.0) GO TO 202
      I=I+1
      JBAD=JBAD+1
      IF(I-JF) 201,205,205
202   IF(I.EQ.1) GO TO 206
      I=I-1
      JF=JF-I
      DO 203 K=1,JF
      XF(K)=XF(K+I)
203   YF(K)=YF(K+I)
      I=I+1
      PRINT 210,I
      GO TO 209
205   JF=1
      PRINT 211
      GO TO 209
206   PRINT 212
209   PRINT 213,JF
      RETURN
210   FORMAT(1X,I10,* POINTS HAD NEGATIVE TIMES*)
211   FORMAT(1X,* ALL POINTS HAD NEGATIVE TIMES*)
212   FORMAT(1X,* NO POINTS HAD NEGATIVE TIMES*)
213   FORMAT(1X,I10,* POINTS WILL BE USED*)
      END
```

APPENDIX A

```
      SUBROUTINE FFT(A,NPOW,NSTAR,XNNYQ,ISIGN)
C A = COMPLEX INPUT AND OUTPUT ARRAY.
C NPOW = ANYTHING. IT IS NOT USED AND IT IS KEPT TO MAKE THE CALL
C                 COMPATIBLE WITH A PREVIOUS VERSION OF FFT.
C NSTAR = NUMBER OF POINTS IN ARRAY.
C XNYQ = INCREMENT IN THE INDEPENDENT VARIABLE.
C ISIGN = SIGN IN THE EXPONENTIAL OF THE FOURIER TRANSFORM.
      COMPLEX A(NSTAR)
      CALL FFT1D(A,NSTAR,ISIGN)
      DO 10 I=1,NSTAR
      A(I)=A(I)*XNYQ
   10 CONTINUE
      RETURN $ END

           IDENT FFT1D
         ENTRY FFT1D
         EXT SIN.,COS.
         VFD 30/5HFFT1D,30/3
FFT1D    BSS 1
         SX7 A0
         SA7 SAVEA0
         SA2 A1+1
         SA3 A1+2
         SB1 X1
         SB2 X2
         SB3 X3
         SA2 B2
         LX2 1                    .N=2*NN
         BX6 X2
         SA6 N
         SB6 X6
         SB4 1
         SA3 B3
         SB7 B4
         BX7 X3
         SA7 ISIGN
         SB1 B1-1
         SX6 B1
         SA6 W
         SB3 B7
         SA0 B7+B7
DO5      GE  B3,B4,S2
         SA1 B1+B4
         SA2 A1+B7
         SA3 B1+B3
         SA4 A3+B7
         BX6 X1
         LX7 X2
         SA6 A3
         SA7 A4
         BX6 X3
         LX7 X4
         SA6 A1
         SA7 A2
S2       SX2 B6
         AX2 1
         SB2 X2
```

```
S3        LE   B4,B2,S5
          SX2 B2
          SB5 A0
          AX2 1
          SB4 B4-B2
          SB2 X2
          GE   B2,B5,S3
S5        SB3 B3+A0
          SB4 B4+B2
          LT   B3,B6,D05
          SX6 A0
          SA6 MMAX
S6        SB7 X6
          SA2 N
          SB6 X2
          GE   B7,B6,S10
          SA1 TWOPI
          SA2 ISIGN
          PX2 B0,X2
          PX6 B0,X6
          DX3 X6*X2
          UX3 X3
          PX4 B0,X3
          NX5 X4
          RX6 X1/X5
          SA6 THETA
          SA1 THETA
          RJ   SIN.
          FX1 X6+X6
          NX7 X1
          SA7 WST
          SX0 B0
          FX6 X0-X6
          NX7 X6
          SA7 WIO
          SA1 THETA
          RJ   COS.
          SA6 WRO
          SA1 ONE
          BX2 X2-X2
          SA3 MMAX
          SB5 X3
          LX3 1
          SB6 X3
          SB7 1
          SA5 W
          SB1 X5
          SA3 N
          SB4 X3
          SA0 B7
D09       SB2 B1+B5
          SB3 A0
D08       SA3 B2+B3
          SA4 A3+B7
          RX5 X1*X3
          RX6 X2*X4
          RX0 X5-X6
```

```
         NX7 X0
         RX5 X2*X3
         RX6 X1*X4
         SA3 B1+B3
         SA4 A3+B7
         RX5 X5+X6
         NX0 X5
         RX5 X3-X7
         NX6 X5
         RX5 X3+X7
         SA6 B2+B3
         NX7 X5
         RX5 X4-X0
         SA7 B1+B3
         SB3 B3+B6
         NX6 X5
         RX0 X4+X0
         SA6 A6+B7
         NX7 X0
         SA7 A7+B7
         LT  B3,B4,D08
         SA3 WR0
         SA4 WI0
         SA5 WST
         BX6 X1
         LX7 X2
         RX2 X5*X2
         RX0 X5*X1
         SA0 A0+B7
         SB3 A0+B7
         RX3 X3-X2
         NX1 X3
         SA0 B3
         SA6 A3
         RX5 X4+X0
         NX2 X5
         SA7 A4
         LT  B3,B5,D09
         SX6 B6
         SA6 NMAX
         EQ  S6
S10      SA1 SAVEA0
         SA0 X1
         EQ FFT10
N        BSS 1
M        BSS 1
ISIGN    BSS 1
NMAX     BSS 1
TWOPI DATA  6.2831853071 7958
THETA    BSS 1
WST      BSS 1
WIO      BSS 1
WRO      BSS 1
ONE      DATA 1.0
SAVEA0   BSS 1
         END
```

```
      SUBROUTINE SMUZ(X,N,M1,M2,FRAC)
C THIS SUBROUTINE  AVERAGES OUT A FUNCTION BETWEEN PEAKS.
C X IS AN ARRAY OF N EQUISPACED ORDINATES.
C M1 OR M2 ARE INTEGERS THAT GIVE A MAXIMUM NUMBER OF POINTS
C BETWEEN TWO MAXIMA OR TWO MINIMA  FOR THE AVERAGING ACTION TO TAKE PLACE.
C FRAC IS THE FRACTION OF THE POINTS FOR WHICH M1 IS USED,
C THE REMAINDER USES M2.
C THE PROCEDURE IS REPEATED UNTIL NO CHANGES ARE INTRODUCED ANYWHERE,
C THE PRINTOUT STATES THE NUMBER OF PASSES OF SMUZ THAT WERE REQUIRED.
      DIMENSION X(N)
      NT=0
10    KFLAG=0
      NT=NT+1
      IFLAG=0
      N1=3   $  N2=N*FRAC    $  M=M1
      IF(FRAC.NE.0.) GO TO 15
      N2=N    $    M=M2
15    XI1=X(1)
      XI=X(2)
      IF(XI.GT.XI1) GO TO 20
      LX=-1
      X2MAX=XI1
      K2MAX=1
      K2MIN=-M
      GO TO 30
20    LX=1
      X2MIN=XI1
      K2MIN=1
      K2MAX=-M
30    XI1=XI
40    DO 1000 I=N1,N2
      XI=X(I)
      IF(XI.GT.XI1) GO TO 60
      IF(LX.EQ.-1) GO TO 900
      LX=-1
      IF(I-K2MAX.LE.M) GO TO 100
      X2MAX=XI1
      K2MAX=I-1
      GO TO 990
60    IF(LX.EQ.1) GO TO 900
      LX=1
      IF(I-K2MIN.LE.M) GO TO 200
      X2MIN=XI1
      K2MIN=I-1
      GO TO 990
100   K1MAX=K2MAX
      X1MAX=X2MAX
      K2MAX=I-1
      X2MAX=XI1
      IF(IFLAG.EQ.1) GO TO 150
      KFLAG=1
      IFLAG=1
      K1MIN=K2MIN-M
      IF(K1MIN.LE.0) K1MIN=1
      X1MIN=X(K1MIN)
      IF(K2MAX.EQ.1) GO TO 150
      CALL AVRG(X1MIN,K1MIN,X2MIN,K2MIN,X1MIN,K1MIN,X1MAX,K1MAX,X,
     1 K1MIN+1,K1MAX)
150   CALL AVRG(X1MAX,K1MAX,X2MAX,K2MAX,X1MIN,K1MIN,X2MIN,K2MIN,X,
     1 K1MAX+1,K2MIN)
      GO TO 990
200   K1MIN=K2MIN
      X1MIN=X2MIN
      K2MIN=I-1
      X2MIN=XI1
      IF(IFLAG.EQ.1) GO TO 250
      KFLAG=1
      IFLAG=1
      K1MAX=K2MAX-M
      IF(K1MAX.LE.0) K1MAX=1
      X1MAX=X(K1MAX)
      IF(K1MIN.EQ.1) GO TO 250
      CALL AVRG(X1MAX,K1MAX,X2MAX,K2MAX,X1MAX,K1MAX,X1MIN,K1MIN,X,
     1 K1MAX+1,K1MIN)
250   CALL AVRG(X1MAX,K1MAX,X2MAX,K2MAX,X1MIN,K1MIN,X2MIN,K2MIN,X,
     1 K1MIN+1,K2MAX)
      GO TO 990
900   IF(IFLAG.EQ.0) GO TO 990
      IF(LX.EQ.1) GO TO 950
      IF(I-K2MIN.LT.M) GO TO 990
      CALL AVRG(X1MAX,K1MAX,X2MAX,K2MAX,X2MIN,K2MIN,XI,I,X,
     1 K2MIN+1,K2MAX)
      CALL AVRG(X2MIN,K2MIN,XI,I, X2MAX,K2MAX,XI,I,X,K2MAX+1,I-1)
      GO TO 980
950   IF(I-K2MAX.LT.M) GO TO 990
      CALL AVRG(X2MAX,K2MAX,XI    ,I     ,X1MIN,K1MIN,X2MIN,K2MIN,X,
     1 K2MAX+1,K2MIN)
      CALL AVRG(X2MAX,K2MAX,XI,I,X2MIN,K2MIN,XI,I,X,K2MIN+1,I-1)
980   IFLAG=0
990   XI1=XI
1000  CONTINUE
      IF(N2.EQ.N) GO TO 1010
      N1=N2+1   $  N2=N   $  N=N2
      GO TO 40
1010  IF(IFLAG.EQ.0) GO TO 1060
      IF(LX.EQ.1) GO TO 1050
      CALL AVRG(X1MAX,K1MAX,X2MAX,K2MAX,X1MIN,K1MIN,X2MIN,K2MIN,X,
     1 K1MIN+1,N)
      GO TO 1060
1050  CALL AVRG(X1MAX,K1MAX,X2MAX,K2MAX,X1MIN,K1MIN,X2MIN,K2MIN,X,
     1 K1MAX+1,N)
1060  IF(KFLAG.EQ.1) GO TO 10
      PRINT 1,     N1,N2, FRAC,NT
1     FORMAT(1X,"N1=",I3,"    N2=",I3,"     FRAC=",F5.2,
     1 "     NUMBER OF PASSES OF SMUZ IS",I3)
      RETURN
      END

      SUBROUTINE AVRG(X1,K1,X2,K2,X3,K3,X4,K4,X,KI,KF)
C THIS SUBROUTINE SUBSTITUTES POINTS IN THE ARRAY X BETWEEN KI AND KF
C BY THE AVERAGE BETWEEN THE TWO STRAIGHT LINES DEFINED BY X1,X2
C AND X3,X4 AT POINTS K1, K2, K3, K4 RESPECTIVELY
      DIMENSION X(1)
      A1=.5/(K2-K1)
      A2=X2*A1
      A1=X1*A1
      B1=.5/(K4-K3)
      B2=X4*B1
      B1=X3*B1
      C1=A2-A1+B2-B1
      C2=A1*K2-A2*K1+B1*K4-B2*K3
      DO 10 J=KI,KF
      X(J)=C1*J+C2
10    CONTINUE
      RETURN
      END
```